From Predictable Paths to Dynamic Racing:  
Transforming Kart Games with Unity ML-Agents

Bui Hoang Ha Chau

# **Introduction**

## **Problem**

In recent years, the advancements in artificial intelligence (AI) and machine learning (ML) have revolutionized various sectors, including the video game industry (1). The application of ML in game development has not only enhanced the gameplay experience but has also introduced novel approaches to game design and interaction. Among various genres, racing games, particularly kart racing games, stand out as an area ripe for innovation through AI.

Traditional kart racing games often rely on pre-programmed paths and simple obstacle avoidance algorithms for non-player character (NPC) behavior. While this approach ensures a functional and predictable gaming experience, it lacks the dynamism and adaptability that could be achieved through more advanced AI techniques. In comparison, modern advancements in ML, such as imitation learning, offer the potential to significantly enhance the complexity and realism of NPC behaviors, making them more competitive and unpredictable (2).

The issue at hand involves applying ML-Agents, Unity's open-source project that enables games and simulations to serve as environments for training intelligent agents, to a kart racing game. The focus is particularly on utilizing and improving imitation learning to create AI that not only mimics human player strategies but also adapts and evolves, providing a more challenging and engaging experience for players. This study aims to address the gap between traditional AI approaches in kart racing games and the possibilities that ML-Agents and imitation learning present, demonstrating how these technologies can transform the gameplay experience.

In addition to the application of imitation learning, this study will also harness the capabilities of Reinforcement Learning (RL) provided by ML-Agents to further enhance our AI models. RL allows agents to learn optimal behaviors through systematic trial-and-error by interacting with the environment, receiving rewards or penalties based on their actions. This approach is particularly suited to the dynamic and ever-changing scenarios presented in racing games, where adaptability and decision-making are crucial. By integrating RL, we aim to develop AI that not only replicates human-like strategies but also innovatively adapts to new challenges and competitor tactics, making the gameplay experience not only realistic but also continuously engaging. The use of RL will complement our imitation learning framework, creating a robust model where AI can effectively learn both from direct human inputs and independent interaction with the game environment, thereby bridging the current gap in AI capabilities within kart racing games.

## **Related Works**

The incorporation of AI through Unity's ML-Agents in racing games has garnered attention for its potential to elevate game mechanics and player experience. The following contributions illustrate this trend:

1. **General Use of ML-Agents in Game Development**: Juliani et al. (3) provided a comprehensive introduction to ML-Agents, showcasing its capabilities in creating complex AI behaviors across various game genres, including racing games. This foundational work lays the groundwork for understanding how ML can be integrated into game development.
2. **Reinforcement Learning with Unity**: Lange et al. (4) discussed the application of reinforcement learning (RL) within Unity, offering insights into how RL techniques, including algorithms like PPO and DQN, can be applied to train agents in simulated environments, relevant to racing game scenarios.
3. **Adaptive NPC Behavior in Racing Games**: A study by Togelius et al. (5) explored the procedural generation of racing tracks and adaptive NPC behavior, providing context on how ML-Agents could be utilized for dynamic difficulty adjustment and enhancing NPC competitiveness in racing games.
4. **Multi-Agent Systems for Competitive Racing**: A review by Shao et al. (6) on multi-agent systems in competitive environments outlines the potential for applying ML-Agents to train multiple AI racers, fostering complex interactions and strategies that can mimic or surpass human players' skills.
5. **Curriculum Learning in Game AI Training**: Justesen et al. (7) demonstrated the effectiveness of curriculum learning in progressively training game AI, a concept that can be directly applied to training racing game AI with ML-Agents to master various tracks and racing conditions.

These references highlight the diverse applications of ML-Agents in racing games, from training sophisticated AI opponents to generating dynamic game environments. They underscore the potential of ML to transform racing games into more engaging and challenging experiences.

# Imitation Learning

Imitation Learning is a subset of machine learning techniques where models learn to perform tasks by mimicking expert behavior, rather than through trial-and-error interactions with the environment. In the context of Unity's ML-Agents, imitation learning provides a framework for developers to train AI agents using recorded data from human players, essentially teaching the AI to replicate human strategies and actions. This approach has been increasingly applied in various domains, including autonomous vehicles, robotics, and, notably, video game development, especially in complex games like racing or kart racing titles (8).

The core idea behind imitation learning is straightforward: leverage the knowledge and expertise of human players to quickly instill advanced skills and behaviors in AI agents. By observing and replicating the actions of human experts, AI can achieve a level of performance that might take significantly longer to reach through traditional reinforcement learning techniques. This is particularly valuable in racing games, where mastering the intricacies of track navigation, speed control, and strategic maneuvering is crucial for competitive gameplay.

In Unity's ML-Agents, imitation learning is implemented through Behavioral Cloning and Generative Adversarial Imitation Learning (GAIL). Behavioral Cloning directly maps observed states to actions, enabling agents to learn from a dataset of state-action pairs collected from human gameplay. GAIL, on the other hand, involves training an agent to perform tasks in a way that is indistinguishable from that of a human expert, using a discriminative model to guide the learning process (9).

The application of imitation learning in game development, particularly with ML-Agents, offers a promising avenue for creating more sophisticated, human-like AI opponents and partners. However, as with any technology, there are benefits and limitations to its use.

## **Advantages**

1. **Accelerated Mastery**: Through imitation learning, AI entities rapidly acquire intricate maneuvers by observing expert gameplay, notably in complex environments such as racing or kart racing games. This method offers a shortcut to assimilating advanced driving techniques, like optimal pathfinding and precision turning, which might be slower to develop via conventional reinforcement learning (10).
2. **Realistic Opponent Behavior**: AI driven by imitation learning, trained on human gameplay, tends to exhibit behaviors that closely mirror human players. This realism enhances player engagement by providing a more formidable and unpredictable challenge, elevating the overall gaming experience (11).
3. **Streamlined Learning Process**: Compared to reinforcement learning, the imitation approach to training AI is generally more resource-efficient, leveraging existing datasets of expert gameplay to expedite the learning phase with reduced computational demands (12).

## **Disadvantages**

1. **Adaptation Limitations**: While effective in cloning expert strategies, imitation learning may not equip AI with the flexibility to navigate novel scenarios that diverge from the training examples. This could result in AI opponents exhibiting predictable, and thus exploitable, behaviors (13).
2. **Data Quality Dependence**: The effectiveness of this learning model is contingent on the richness and representativeness of the training data. Inferior quality data can lead to less optimal or flawed AI behaviors, detracting from the authenticity of the gaming experience (14).
3. **Potential for Overfitting**: There exists a risk of AI over-specializing to the scenarios present in the training dataset, compromising its ability to generalize its skills across diverse racing tracks or under varying conditions not encountered during its training (15).

Imitation learning serves as a rapid conduit to proficient behavior within racing games, contrasting with the exploratory nature of reinforcement learning, which fosters AI's development of innovative strategies through reward-based optimization. However, imitation learning's reliance on high-quality input data and its potential for reduced adaptability highlight critical considerations in choosing the appropriate AI training methodology for racing games.

## **Imitation Learning and PPO combinations**

Integrating Imitation Learning with Proximal Policy Optimization (PPO) in ML-Agents for a racing game is a strategic approach that harnesses the synergies of both learning techniques. This fusion method aims to produce AI agents that not only quickly grasp the essentials of racing through human emulation but also continuously evolve their strategies through environmental interaction.

### **Synergistic Approach**

* **Foundation with Imitation Learning**: Start the AI’s journey by immersing it in imitation learning, where the agents digest a wealth of human gameplay data. This phase is crucial for embedding the agents with a solid base of racing knowledge, including mastering complex maneuvers, and understanding diverse racing strategies directly from human expertise.
* **Strategic Evolution with PPO**: With a foundational knowledge base established, the AI is then elevated through PPO training. This stage is about refining the AI's decision-making processes and encouraging the development of innovative strategies by experimenting within the game's ecosystem. PPO's strength in optimizing policies through feedback loops makes it an excellent tool for enhancing the AI's adaptive skills and problem-solving capabilities.

### **Execution Strategy**

1. **Imitation Learning Initiation**: Begin with compiling and utilizing a dataset of high-quality human gameplay to teach the AI the rudiments of racing. This initial training sets a high baseline of performance and ensures that the AI exhibits human-like behavior from the outset.
2. **Progressive Transition to PPO**: Gradually introduces the AI to PPO training, focusing on expanding its tactical repertoire and honing its racing acumen. This transition is key to unlocking the AI's potential for self-improvement and adaptation beyond the initial imitative behaviors.
3. **Iterative Refinement and Evaluation**: It's essential to maintain an iterative cycle of training, evaluation, and adjustment. By continuously assessing the AI's performance and adjusting the balance between imitation learning and PPO training, developers can fine-tune the AI to achieve optimal performance across various racing scenarios.

By marrying imitation learning with PPO, developers can leverage the rapid skill acquisition of the former and the dynamic strategy optimization of the latter. This integrated approach not only expedites the training process but also ensures that AI agents remain competitive and versatile, capable of facing both the predictable elements of racing games and the unpredictable strategies of human opponents.

# **Methodology**

The methodology section outlines the systematic approach adopted for integrating Imitation Learning with Proximal Policy Optimization (PPO) in developing sophisticated AI for a racing game using Unity's ML-Agents. This fusion aims to combine the rapid skill acquisition of imitation learning with the dynamic strategy optimization of PPO, thereby creating AI agents that exhibit both high levels of performance and adaptability in complex racing environments. The process is structured into several key phases, each contributing uniquely to the development of AI agents capable of competing against human players and adapting to varied racing conditions.

## **Preparing Environment**

Create a straightforward racetrack devoid of obstacles or intricate features, accommodating 12 karts simultaneously racing. The course should follow a linear path, starting from the left and concluding after one lap. With 12 initial starting positions, karts should respawn randomly at the end of each episode. Install checkpoints along the track, beginning from the nearest starting point and concluding at the starting position. All checkpoints should be compiled into a list, and rewards should only be granted if the current target index of a kart aligns with the index of a checkpoint on the list. Additionally, implement a dead zone collider beneath the track to prevent karts from moving at high speeds and veering off the road.

Additionally, to facilitate quicker learning, the road has been categorized into various types: straight stretches, left curves, right curves, zigzags, and a segment emulating train speed for the fastest times. These segments are then amalgamated into one continuous road to optimize performance.

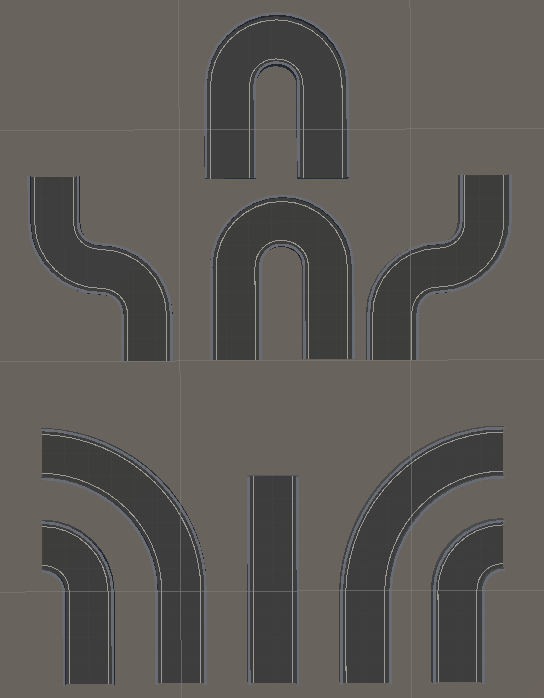


Fig 1: Road Types

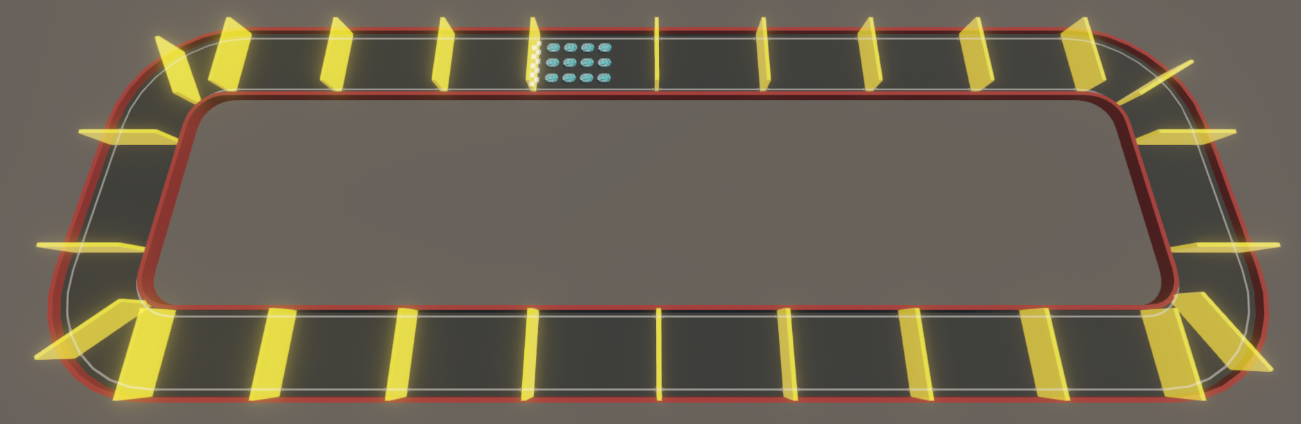


Fig 2: Simple Road

The kart operates similarly to a standard four-wheel car, capable of moving forward, backward, turning left, and turning right. However, to simplify the task, the agent is deprived of the ability to move backward, and its primary objective is to navigate while minimizing collisions with walls.

## **Learning Environment**

The game offers players the opportunity to control the kart from three distinct viewpoints: inside the kart itself, from a third-person perspective, and from a viewpoint positioned in front of the car. Players can interact using input devices such as keyboards or controllers. Conversely, the kart agent perceives its environment solely through positional data and distances provided by the Unity Engine. It's advisable to normalize all observed vectors by the agents to adhere to best practices when employing neural networks.

Throughout the experiments, a reward function has been devised to incentivize the kart agent's behavior towards winning. This function encompasses various factors such as collisions with walls or other karts, as well as the time taken to move between two checkpoints, and instances where the kart times out and cannot reach the next checkpoint. The aim is to maximize this reward function, which guides the agent towards sequences of actions leading to victory while discouraging detrimental behaviors through punishments.

|  |  |  |
| --- | --- | --- |
| **Name** | **Reward** | **Description** |
| Reach next check point | 0.5 / total check point count | Reward when reach right check point |
| Reach last check point | 0.5 | Reward when finish race |
| Hit wall | -0.1 | When hit the wall |
| Stay at wall | -0.01 | When kart still hit wall |
| Time out | -1 | When time to reach next check point is time out |
| Collect observation | -0.001 | Each second when collect observation |

Table 1: Reward Points

## **Learning Environment**

* + 1. **Learning with PPO**

PPO proved to be a robust choice, training at a moderate pace and stabilizing towards the end of the training process. However, it faced challenges in navigating curves and complex segments of the track, showing a need for improvements in dynamic environments.

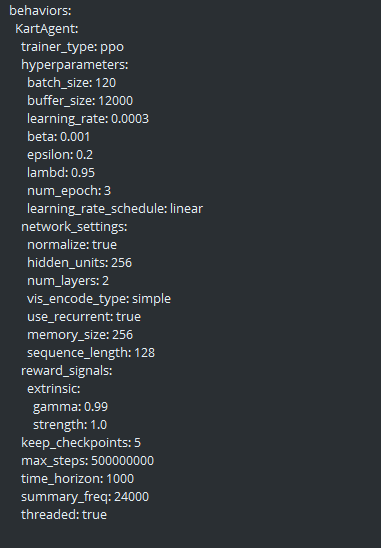


Fig 3: PPO behaviors config

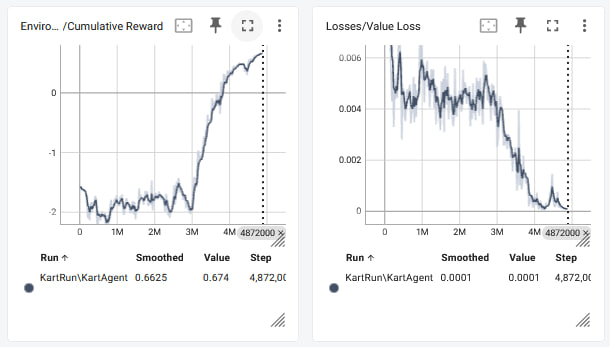


Fig 4: PPO results

* + 1. **Learning with POCA**

POCA displayed instability and required significant training time, reflecting its complexity and the computational effort involved in achieving satisfactory results.

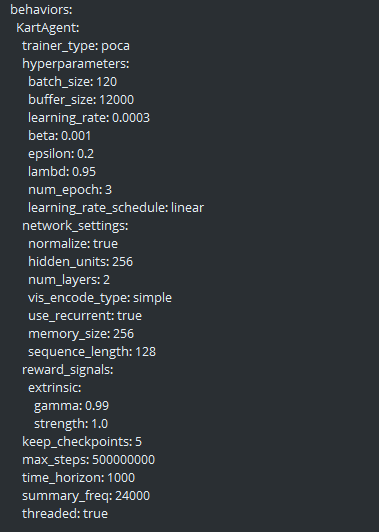


Fig 5: POCA behaviors config

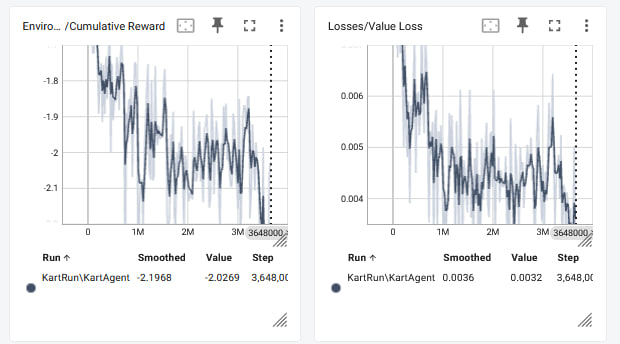


Fig 6: POCA result

* + 1. **Improvement of POCA**

Enhancements made by POCA led to more stable performance and reduced training time. These adjustments yielded better outcomes, yet there was still room for optimization to meet higher standards of efficiency and effectiveness.

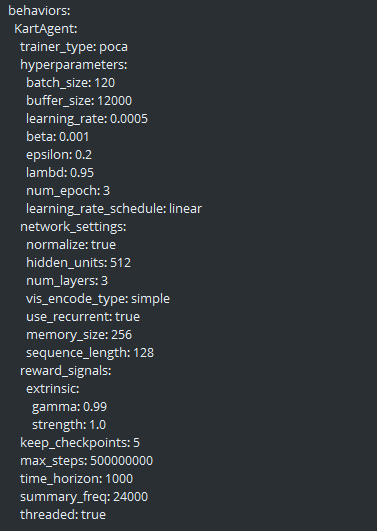


Fig 7: Improvement of POCA behaviors config

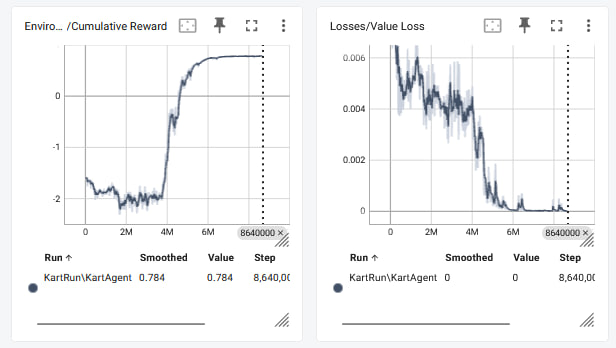


Fig 8: Improvement of POCA result

* + 1. **Learning with PPO and Imitation Learning**

**Integrating PPO with Imitation Learning significantly accelerated the training process and improved stability. This combination was particularly effective in handling difficult curves, minimizing crashes and enhancing overall maneuverability.**

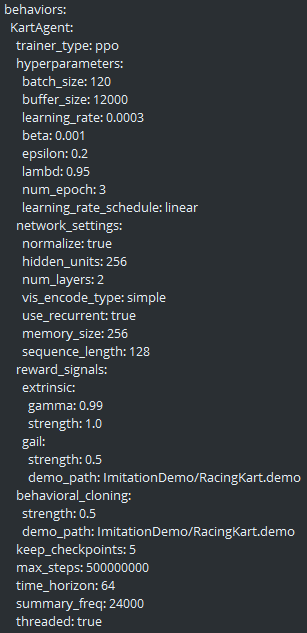


Fig 9: PPO and Imitation Learning behaviors config

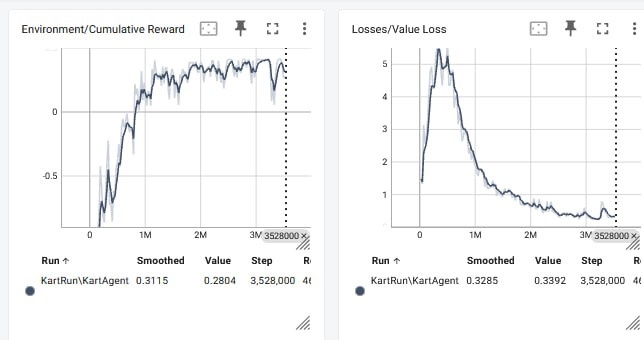


Fig 10: PPO and Imitation Learning result

# **Conclusion**

This study highlights the integration of advanced machine learning techniques, specifically Proximal Policy Optimization (PPO) and Imitation Learning, within Unity's ML-Agents to improve AI in kart racing games. We observed that while PPO provides a solid foundation, it struggles with complex track navigation. Enhancements in Partial Observable Cooperative Agent (POCA) improved stability and training efficiency, but combining PPO with Imitation Learning proved most effective, dramatically enhancing AI adaptability, and reducing training time.

This combined approach allowed the AI to mimic human-like performance, making game interactions more challenging and engaging. Our findings advocate for leveraging sophisticated ML techniques to elevate gameplay realism and complexity, encouraging further exploration at the intersection of AI and game development to continually push the boundaries of interactive entertainment.

# **References**

1. Juliani, Arthur, et al. "Unity: A General Platform for Intelligent Agents." arXiv preprint arXiv:1809.02627 (2018).
2. Ho, Jonathan, and Stefano Ermon. "Generative Adversarial Imitation Learning." Advances in Neural Information Processing Systems. 2016.
3. Juliani, A., et al. "Unity: A General Platform for Intelligent Agents." arXiv preprint arXiv:1809.02627 (2018).
4. Lange, S., et al. "A comprehensive introduction to reinforcement learning and its application in Unity." Procedia Computer Science, 2019.
5. Togelius, J., et al. "The Racing Game AI Playbook: Making Racing Games More Challenging and Fun." Game Developer, 2020.
6. Shao, K., et al. "Review on the Application of Artificial Intelligence in Racing Games." Entertainment Computing, 2021.
7. Justesen, N., et al. "Automated Curriculum Learning for Neural Networks." IEEE Conference on Games (CoG), 2019.
8. Ho, Jonathan, and Stefano Ermon. "Generative Adversarial Imitation Learning." *Advances in Neural Information Processing Systems*. 2016.
9. Juliani, Arthur, et al. "Unity: A General Platform for Intelligent Agents." *arXiv preprint arXiv:1809.02627* (2018).
10. Sutton, R. S., & Barto, A. G. "Introduction to Reinforcement Learning." MIT Press, 2018.
11. Schaal, S. "Learning from demonstration." *Advances in neural information processing systems*, 1997.
12. Pan, S. J., & Yang, Q. "A Survey on Transfer Learning." *IEEE Transactions on Knowledge and Data Engineering*, 2010.
13. Torabi, F., Warnell, G., & Stone, P. "Behavioral Cloning from Observation." *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.
14. Goodfellow, I., Bengio, Y., & Courville, A. "Deep Learning." MIT Press, 2016.
15. Taylor, M. E., & Stone, P. "Transfer Learning for Reinforcement Learning Domains: A Survey." *Journal of Machine Learning Research*, 2009.