Automating agent decisions in a virtual environment

Uicoabă Alexandru   
*Facultatea de Electronică, Telecomunicații și Tehnologii Informaționale  
Universitatea Politehnica Timișora*Timișoara, România   
alexandru.uicoaba@student.upt.ro

*Abstract*—This paper proposes an analysis of how smart agents are implemented in virtual environments. Video games are meant to introduce the user to a virtual world. The diversity of these virtual worlds is huge, whether we refer to board games, action, racing, or strategy games, with the ability to play in single-player or multi-player mode. Following this analysis, an experiment was performed in wich a kart agent learns to drive in a virtual environment, using several machine learning. Following the results, it is observed that the agent’s performance is given both by the chosen machine learning algorithm and by the virtual environment.

Keywords—arificial intelligence, smart agents, video games, machine learning, neural networks, reinforcement learning

# Introducere

Video games have a long history, from the first games developed on the **Atari** console to the well-known **Pac-Man**, **Super Mario,** and **Sonic**, to contemporary ones, such as **Cyberpunk 2077.**

Intelligent agents are computer systems that are placed in various virtual environments, in this case, virtual worlds, which are capable of autonomous actions to achieve well-defined objective.[1] They aim to solve problems in the game, which could be defeating an opponent in battle or navigating a maze.[2]

Virtual worlds within video games are a playground for a wide variety of algorithms to try new things. Depending on the results obtained, these ideas can be transferred and applied in real life. The goal of artificial intelligence is to create intelligent agents who can make the best decisions, both in virtual environments and in the real world..[3]

To train these agents, existing games can be used, or games can be created from scratch, in a multitude of programming languages, to create the environment we need.

In the last decade, more and more work has been done in this direction, due to the technological advancement we are in, being advanced hardware components for the use of developed algorithms that need high computing power. Thus, there have been games in which the graphics are getting closer to reality.

So, through smart agents, the emphasis is on getting a more natural behavior of the characters in the games, by using fuzzy logic and neural networks. In this way, we get characters that behave naturally and fit into the game's setting in a more pleasant way.[4]

This leads to the following research questions :

Question 1 - What are the machine learning techniques used for agents?

Question 2 - What types of algorithms are more efficient?

Question 3 - What data does the algorithm need to learn?

Question 4 - Is a trained machine learning agent better than a real person?

# Methodology

This paper is the result of a systematic search for how to automate the decisions of agents in a virtual environment, such as video games. Also, after the works were analyzed, the experiment was performed.

## Machine learning algorithms

Down below is a breakdown of the agents whether they are trained as a result of human intervention or not.

**Unsupervised machine learning** methods are particularly useful in descriptive tasks, as they aim to find relationships in a data structure without having a measured result. This category of machine learning is called unsupervised because it lacks a response variable that can supervise the analysis. [3]

**Supervised machine learning** methods are used to describe predictive tasks because they are intended to predict and classify a particular outcome of interest (if a particular person is prone to certain diseases based on medical information). Supervised learning has been applied to large data structures, because in order to achieve good accuracy we need to "feed" the ML model with a large set of data in the learning cycle. [3]

In this ML category, the data is divided into several sets. A first set is the training set that are introduced in the model. The second set of data is the test set, which determines the performance of the model by calculating several indicators, such as accuracy. Usually the ratio between the two sets is 9 to 1.

In **Reinforcement Learning**, a software agent makes observations and take actions within the environment, and in return it receives rewards. Its objective is to learn to act in a way that will maximize its expected rewards over time.[3]

## Related work

Zpepei Wei and his team have obtained an autonomous agent capable of playing the well-known game Snake. The training of the agent was done by using the Convolutional Neural Network (CNN) with a variety of Q-learning. This option aims to avoid situations where rewards are rare and come late. Experiments have shown that through this model, the agent has even managed to exceed the level reached by people.[12]

# IMPLEMENTATION

This section will present the implementation of the experiment that aims to train an agent in a virtual environment, both by using ML unsupervised algorithms and by using reinforcement learning.

## Colecting data from the virtual environment

The first step is to collect the necessary data of the agent, in this case, the kart, which is necessary for training. As a medium, the microgame kart from Unity was used, which provides the circuit together with the kart. To be able to collect the data, the following is collected through a script: the positions on the three axes, the distance to the object from the 5 sensors, the direction of movement of the kart, the state of the game, and the time in the game.

Access to this information is done by calling a get endpoint of an API made with the Flask API. At each frame of the game, the kart information is updated by a POST call from the unity to the water environment. Another way to access data is through the csv files that are generated for each game.

The data flow, both for the scenario in which the kart learning is performed is done through supervised machine learning methods, and through reinforcement learning are presented in diagrams 1 and 2.

## Unsupervised ML

A first experiment that aims to train the agent to drive in the virtual environment is through supervised algorithms. In this sense, two algorithms will be used, namely Random Forrest and Decision Tree.

Since we are talking about supervised machine learning algorithms, they need a set of data to train on. So, a human player simulated driving the kart for several episodes, the kart data was saved in CSV files, one file for each episode.

After combining the CSV files into one, the data cleaning operations were performed. The next step was to choose the features for the ML algorithms. From the point of view of dividing the data between training data and validation data, a ratio of 9 to 1 was used

After the machine learning model has been trained, each frame of the game transmits the data from the environment through the POST endpoint to the script in Python, the data is taken by the ML model, whether we are talking about Random Forrest or Decision Tree. That decision is sent back to the unity environment as a response to the same endpoint, after which a mapping is done to the values ​​in the Unity for kart control.

The above steps are repeated until the game reaches a finished state, such as when the kart reaches the finish line, or when the time expires.

Its output, which represents the decision that the kart will take in the next step, is an integer value from one to 15, representing all possible combinations for the 4 commands of the kart (forward, backward, left and right).

## Reinforcement Learning

The second experiment involves learning the agent to drive (on the same circuit used for the first experiment) using a reinforcement learning algorithm. This type of algorithm is based on making decisions and observing their effect.

At the time t, a certain decision is made. At time t + 1, it is checked whether the decision was a good one or not. Depending on this, the agent receives a reward or a penalty. The goal is for the agent to receive as many rewards as possible so that they can perform as well as possible in that environment.

In this scenario, in addition to the endpoint for communication between the python script and the Unity environment, two additional endpoints are used. One is used by the reinforcement learning environment to retrieve the latest data from the game, and the other is to send actions provided by the model to the link script, which are then sent to the game.

**OpenAi Gym** was used to create the learning environment using Reinforcement Learning. It has several game scenarios that can be used, or as in this case, can be created from scratch. To achieve the environment, it is necessary to define several specialized functions.

The first is the \_\_init\_\_, which has the initialization part of the game. Another important function is the \_\_reset\_\_ function, which is called when a learning sequence is completed. This can be done either if the kart reaches the finish line or if the time expires.

In the step function, the decision is made and the reward is calculated. Here, too, the information from the kart that is taken into account in the next step is received and the actions taken by it are sent to unity.

In this case, several factors are taken into account in calculating the reward. The first factor is the direction of travel of the kart. if he moves in front, according to the direction of the circuit tour, he will receive a reward, otherwise, he will be penalized. Another criterion depends on the value of the 5 obstacle detection sensors. if the kart approaches an object (currently on the parapet of the circuit), it will receive a penalty that increases as it approaches the obstacle in an almost exponential manner.

Diagram 1

Data flow for a supervised machine learning approach

|  |  |  |  |
| --- | --- | --- | --- |
| **Nume articol** | **Tehnica de învățare** | **Joc** | **Platformă** |
| Real-Time Monte Carlo Tree Search in Ms Pac-Man | Monte Carlo Tree Search | single game | - |
| Analysis of Self-Adaptive Monte Carlo Tree Search in General Video Game Playing | Self-adaptive Monte-Carlo Tree Search  N-Tuple Bandit Evolutionary Algorithm | multi games | - |
| General Video Game AI: A Multitrack Framework for Evaluating Agents, Games, and Content Generation Algorithms | Monte-Carlo Tree Search | multi games | - |
| Autonomous Agents in Snake Game via Deep Reinforcement Learning | Deep Q-learning network | single game | - |
| A MiniMax Agent for Playing Ntxuva Game – The Mozambican Variant of Mancala | Minimax Algorithm | single game | - |
| Improvisation of Minimax Algorithm with Multi Criteria Decision Maker (MCDM) in the Intelligent Agent of Card Battle Game | Minimax Algorithm with Multi Criteria Decision Maker | single game | - |
| Emulating Human Play in a Leading Mobile Card Game | Information Set Monte Carlo Tree Search | single game | - |
| Intelligent online case-based planning agent model for real-time strategy games | Online Case-based Planning Reinforcement Learning | single game | - |
| Rogue-Gym: A New Challenge for Generalization in Reinforcement Learning | Proximal Policy Optimization | single game | - |

Diagram 2

Data flow for a reinforcement learning approach

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nume articol** | **Tehnica de învățare** | **Joc** | **Platformă** | |
| A Gradient-Based Reinforcement Learning Algorithm for Multiple Cooperative Agents | Reinforcement Learning | multi games | | - |
| Multi-agent system application in accordance with game theory in bi-directional coordination network model | Bi-directional coordination network  Recurrent Neural Networks | - | | - |
| Altruism and Selfishness in Believable Game Agents: Deep Reinforcement Learning in Modified Dictator Games | deep reinforcement learning | single game | | - |
| Swarm intelligence for autonomous cooperative agents in battles for real-time strategy games | Markov Decision Process  Reinforcement Learning | single game | | - |
| Catch me if you can: A pursuit-evasion game with intelligent agents in the Unity 3D game environment | A\* algorithm | single game | | - |
| General Video Game AI: A Multitrack Framework for Evaluating Agents, Games, and Content Generation Algorithms | Monte-Carlo Tree Search | multi games | | - |
| A Methodology for Creating Generic Game Playing Agents for Board Games | Monte-Carlo Tree Search  Cascade Correlation Neural Network | single game | | - |
| Utilizing Multiple Agents for Decision Making in a Fighting Game | Deep reinforcement learning | single game | | - |
| Evaluating Competition in Training of Deep Reinforcement Learning Agents in First-Person Shooter Games | Deep Reinforcement Learning | single game | | - |
| Evaluating the Performance of the Deep Active Imitation Learning Algorithm in the Dynamic Environment of FIFA Player Agents | Deep Active Imitation  Direct Imitation Learning | single game | | - |
| Reinforcement Learning with an Extended Classifier System in Zero-sum Markov Games | eXtended Classifier System | single game | | - |
| A Multi-agent Design of a Computer Player for Nine Men's Morris Board Game using Deep Reinforcement Learning | Convolutional Neural Network  Monte Carlo Tree Search | single game | | - |
| Learning to Coordinate with Deep Reinforcement Learning in Doubles Pong Game | Deep Q Networks | single game | | - |

# RESULTS

After conducting the two experiments and collecting the related data, in case the route is known, the following results were obtained.

It can be observed first of all, by observing the time taken by the kart in each scenario to reach the finish line. In this comparison, the time obtained by a real person, karting through the Random Forrest and Decision Tree algorithm after five repetitions, was taken into account, taking into account only the best time.

As can be seen in Fig. 1, the differences between the three scenarios do not exist. This is also because the circuit is quite simple, with an oval configuration, similar to those in the NASCAR.

For the reinforcement learning part, the time taken is not close to that of the other algorithms, because the kart learns from scratch how to behave, compared to other algorithms in which the kart makes a decision similar to that of the person in a similar situation.

Fig. 1. Human vs Decission Tree vs Random Forrest

This experiment also shows that if the route in which the kart is to be learned is known, the use of prediction algorithms is more optimal. On the other hand, if the route is not known, or if it is changed or generated randomly, it is preferable to use reinforcement learning algorithms.

### Limitation

Având în vedere că în acest articol științific au fost introduse doar articole provenite din baza electronică a celor de la IEEE, este posibil ca unele articole să fie omise. Pe viitor se dorește extinderea articolelor analizate, prin introducerea de articole provenite și din alte baze de date electronice, diferite de IEEE.

# Conclusion

In this paper, an experiment was performed to implement two scenarios for learning an agent to drive in a virtual environment, using supervised machine learning algorithms, as well as reinforcement learning.

In the future, the complexity of the experiment can be continued in several directions. One direction would be to switch from a single agent to a multi-agent mode and observe how they interact with each other. Another direction would be to use several circuit tours, and to observe the differences in time and route between them. One last direction could be to increase the complexity of the circuit and change it in a random way, and to observe how the agent adapts to the change.

##### Reference

[1] L. Padgham and M. Winikoff, Developing Intelligent Agent Systems: A Practical Guide, 1st ed. Wiley, 2004.

[2] F. Safadi, R. Fonteneau, and D. Ernst, ‘Artificial intelligence in video games: Towards a unified framework’, Int. J. Comput. Games Technol., vol. 2015, 2015.

[3] D. A. Chentsov and S. A. Belyaev, ‘Monte Carlo Tree Search Modification for Computer Games’, in 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Jan. 2020, pp. 252–255, doi: 10.1109/EIConRus49466.2020.9039281.

[4] F. Dignum, J. Westra, W. A. van Doesburg, and M. Harbers, ‘Games and Agents: Designing Intelligent Gameplay’, Int. J. Comput. Games Technol., vol. 2009, pp. 1–18, 2009, doi: 10.1155/2009/837095.

[5] K. Petersen, S. Vakkalanka, and L. Kuzniarz, ‘Guidelines for conducting systematic mapping studies in software engineering: An update’, Inf. Softw. Technol., vol. 64, pp. 1–18, Aug. 2015, doi: 10.1016/j.infsof.2015.03.007.

[6] T. Pepels, M. H. M. Winands, and M. Lanctot, ‘Real-Time Monte Carlo Tree Search in Ms Pac-Man’, IEEE Trans. Comput. Intell. AI Games, vol. 6, no. 3, pp. 245–257, Sep. 2014, doi: 10.1109/TCIAIG.2013.2291577.

[7] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, ‘General Video Game AI: A Multitrack Framework for Evaluating Agents, Games, and Content Generation Algorithms’, IEEE Trans. Games, vol. 11, no. 3, pp. 195–214, Sep. 2019, doi: 10.1109/TG.2019.2901021.

[8] C. F. Sironi and M. H. M. Winands, ‘Analysis of Self-Adaptive Monte Carlo Tree Search in General Video Game Playing’, in 2018 IEEE Conference on Computational Intelligence and Games (CIG), Aug. 2018, pp. 1–4, doi: 10.1109/CIG.2018.8490402.

[9] H. Baier, A. Sattaur, E. J. Powley, S. Devlin, J. Rollason, and P. I. Cowling, ‘Emulating Human Play in a Leading Mobile Card Game’, IEEE Trans. Games, vol. 11, no. 4, pp. 386–395, Dec. 2019, doi: 10.1109/TG.2018.2835764.

[10] Y. Kanagawa and T. Kaneko, ‘Rogue-Gym: A New Challenge for Generalization in Reinforcement Learning’, in 2019 IEEE Conference on Games (CoG), Aug. 2019, pp. 1–8, doi: 10.1109/CIG.2019.8848075.

[11] I. Fathy, M. Aref, O. Enayet, and A. Al-Ogail, ‘Intelligent online case-based planning agent model for real-time strategy games’, in 2010 10th International Conference on Intelligent Systems Design and Applications, Nov. 2010, pp. 445–450, doi: 10.1109/ISDA.2010.5687225.

[12] Z. Wei, D. Wang, M. Zhang, A. Tan, C. Miao, and Y. Zhou, ‘Autonomous Agents in Snake Game via Deep Reinforcement Learning’, in 2018 IEEE International Conference on Agents (ICA), Jul. 2018, pp. 20–25, doi: 10.1109/AGENTS.2018.8460004.

[13] F. D. M. A. Ali, E. Gimo, and S. M. Saide, ‘A MiniMax Agent for Playing Ntxuva Game – The Mozambican Variant of Mancala’, in 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Aug. 2020, pp. 1–5, doi: 10.1109/icABCD49160.2020.9183848.

[14] S. D. H. Permana, K. B. Y. Bintoro, B. Arifitama, A. Syahputra, and M. Cendana, ‘Improvisation of Minimax Algorithm with Multi Criteria Decision Maker (MCDM) in the Intelligent Agent of Card Battle Game’, in 2019 International Conference on Electrical Engineering and Computer Science (ICECOS), Oct. 2019, pp. 254–258, doi: 10.1109/ICECOS47637.2019.8984477.

[15] Z. Zhang, D. Wang, D. Zhao, Q. Han, and T. Song, ‘A Gradient-Based Reinforcement Learning Algorithm for Multiple Cooperative Agents’, IEEE Access, vol. 6, pp. 70223–70235, 2018, doi: 10.1109/ACCESS.2018.2878853.

[16] D. Daylamani-Zad and M. C. Angelides, ‘Altruism and Selfishness in Believable Game Agents: Deep Reinforcement Learning in Modified Dictator Games’, IEEE Trans. Games, pp. 1–1, 2020, doi: 10.1109/TG.2020.2989636.

[17] Y. Takano, S. Ito, T. Harada, and R. Thawonmas, ‘Utilizing Multiple Agents for Decision Making in a Fighting Game’, in 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), Oct. 2018, pp. 594–595, doi: 10.1109/GCCE.2018.8574675.

[18] P. B. S. Serafim, Y. L. B. Nogueira, C. A. Vidal, and J. B. C. Neto, ‘Evaluating Competition in Training of Deep Reinforcement Learning Agents in First-Person Shooter Games’, in 2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Oct. 2018, pp. 117–11709, doi: 10.1109/SBGAMES.2018.00023.

[19] J. Abukhait, A. Aljaafreh, and N. Al-Oudat, ‘A Multi-agent Design of a Computer Player for Nine Men’s Morris Board Game using Deep Reinforcement Learning’, in 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), Oct. 2019, pp. 489–493, doi: 10.1109/SNAMS.2019.8931879.

[20] C. Wang, H. Chen, C. Yan, and X. Xiang, ‘Reinforcement Learning with an Extended Classifier System in Zero-sum Markov Games’, in 2019 IEEE International Conference on Agents (ICA), Oct. 2019, pp. 44–49, doi: 10.1109/AGENTS.2019.8929148.

[21] E. A. O. Diallo, A. Sugiyama, and T. Sugawara, ‘Learning to Coordinate with Deep Reinforcement Learning in Doubles Pong Game’, in 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec. 2017, pp. 14–19, doi: 10.1109/ICMLA.2017.0-184.

[22] J. Zhang, G. Wang, S. Yue, Y. Song, J. Liu, and X. Yao, ‘Multi-agent system application in accordance with game theory in bi-directional coordination network model’, J. Syst. Eng. Electron., vol. 31, no. 2, pp. 279–289, Apr. 2020, doi: 10.23919/JSEE.2020.000006.

[23] M. Rezende and L. Chaimowicz, ‘A Methodology for Creating Generic Game Playing Agents for Board Games’, in 2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Nov. 2017, pp. 19–28, doi: 10.1109/SBGames.2017.00011.

[24] M. P. P. Faria, R. M. S. Julia, and L. B. P. Tomaz, ‘Evaluating the Performance of the Deep Active Imitation Learning Algorithm in the Dynamic Environment of FIFA Player Agents’, in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Dec. 2019, pp. 228–233, doi: 10.1109/ICMLA.2019.00043.

[25] İ. Şahin and T. Kumbasar, ‘Catch me if you can: A pursuit-evasion game with intelligent agents in the Unity 3D game environment’, in 2020 International Conference on Electrical Engineering (ICEE), Sep. 2020, pp. 1–6, doi: 10.1109/ICEE49691.2020.9249828.

[26] D. Daylamani-Zad, L. B. Graham, and I. T. Paraskevopoulos, ‘Swarm intelligence for autonomous cooperative agents in battles for real-time strategy games’, in 2017 9th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games), Sep. 2017, pp. 39–46, doi: 10.1109/VS-GAMES.2017.8055809.