International Journal of Innovative Research in Engineering
Volume 3, Issue 3 (May-June 2022), PP: 437-439.
www.theijire.com **ISSN No: 2582-8746**

Automating Pacman with Deep Q-Learning Using PYgame

G.VIJAY KUMAR[1], N. SAI KUMAR[2], U.VARSHA[3], SK.AFROZ[4]
[1]*Assistant Professor, Computer Science and Engineering, CMR Technical Campus, Hyderabad, India.*
[2,3,4]*Student, Computer Science and Engineering, CMR Technical Campus, Hyderabad, India.*

How to cite this paper:
G.VIJAY KUMAR[1], N. SAI KUMAR[2], U.VARSHA[3], SK.AFROZ[4], "Automating Pacman with Deep Q-Learning Using PYgame", IJIRE-V3I03-437-439.

Abstract: *A computer program capable of doing a human-level performance on a number of games. Just like a human, the algorithm played based on its vision of the screen. Starting from scratch, it discovered gameplay strategies that let it meet (and in many cases, exceed) human benchmarks. In the years since researchers have made a number of improvements that super-charge performance and solve games faster than ever before. We've been working to implement these advancements in Keras — the open-source, highly accessible machine learning framework — and in this post, we'll walk through the details of how they work and how they can be used to master Ms. Pac-man.*

KeyWords *: Deep q-learning, Reinforcement Learning, Deep Neural Networks, Markovian decision process.*

I.INTRODUCTION

The Deep Q-learning algorithm and similar algorithms are part of the Reinforcement learning algorithm, a subset of machine learning. In reinforcement learning an agent is under the Environment and based on the actions of the agent it will get the reward. It takes an action, which changes the environment and feeds it the reward associated with that change. Then it takes a look at its new state and settles on its next action, repeating the process endlessly or until the environment terminates. This decision-making loop is more formally known as a *Markov decision process* (MDP). It's easy to see how Atari games like Ms. Pac-man fit into the MDP framework. The player is the Pacman in this environment and moves automatically based on the learning. Every time it has to try to get the best reward and increase its score. and when playing the game it gets the reward and punishment from the environment. Based on the values the movement of the Pacman was decided. The main aim of the Pacman is eating all the dots.

II.LITERATURE SURVEY

Pac-man is a video game, which was originally developed by Namco in 1980 [15]. Since its introduction, countless remakes were released. When playing the game it has to eat dots that character is the Pacman. And it consists of ghosts also in that environment. The grid is called the maze it is filled with dots and when Pacman eats the dots it increases the score. If the Pacman was chased by the ghost when eaten the dots if ghosts are caught up with the Pacman the player looses the game and consequently the end-of-game. After eating special capsules the player has got the opportunity to also eat ghosts (resulting in a positive reward) for a short period of time. These capsules are generally there in the corners of the game maze. Finally, the game is won by eating all dots. The original version of Pac-man behaves in a strict deterministic manner. However, in some later versions of the game, including the implementation used in this study, ghost movement is stochastic. Also, In this Pac-man implementation does not include multiple lives. An agent situated in this environment performs actions in discrete time steps. Every time step, the environment undergoes a stochastic transition of state depending on the action chosen by the agent. Every state transition results in a reward. A completed Pac-man game forms a sequence of state transitions, actions, and rewards. The goal of the Pacman is to find the best actions to maximize the cumulative reward over this sequence**.**

III.PROPOSED METHODOLOGY

An implementation of the Pac-man game needed the relatively easy exchange of states, actions, and rewards between the Pacman game and the algorithm. In addition, to effectively learn a policy using Deep Q-networks a vast amount of iterations will be needed. Hence, it is helpful during training if simulations of the Pac-man game can run faster than in real-time. The DQN learning algorithm is the modified version of the deepQN tensorflow program. The Deep Q-Networks was originally built to play automatically using ATARI games solely the screen pixel values and rewards. It uses in Python libraries, Tensorflow, and OpenCV in combination with the Arcade Learning Environment (ALE). The code dependent on OpenCV and responsible to interact with ALE was removed. Furthermore, the implementation of the gameplay was rewritten to function in combination with the latest versions of Python (3.7) and Tensorflow (0.8r), and the chosen Pac-man implementation

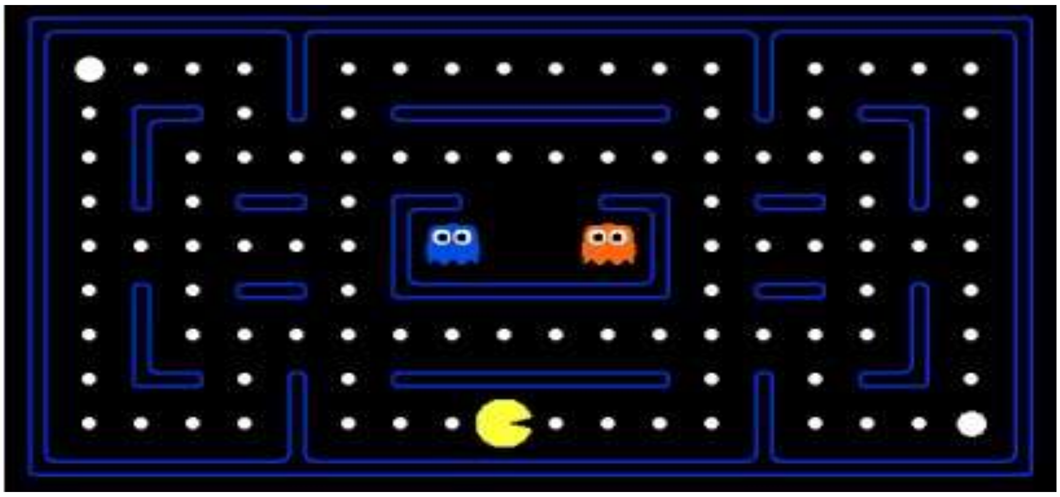

Figure1: Medium Map

The main goal of modeling is to insert as little as possible ’oracle’ knowledge. This means that the agent was learning using the algorithm that environment on its own without explicitly being told how the environment works. Which does not contain any explicit information about the input states. And that found the path move that way. The observations of game states that are presented to the deep Q-learning algorithm should ideally match observations made by a human when playing the game. This study does not use raw pixel values, as previously done, but defines states as a tensor representation of the game grid. By doing this the dimensionality of the input data decreases significantly, but very little ’oracle’ knowledge is told to the algorithm. As stated earlier (Section 1.2) the Markov property in MDPs defines that transitions of state only depend on the present state and that prior history is not relevant. Applied to the Pac-man game this means that does not need any memory to store all past game frames. However, the ghosts in the Pac-man game move in a certain direction, but lack any acceleration. Therefore, the state is defined as the two last game frames as shown in Figure. This way a state includes both positional and directional information. Each game frame consists of a grid or matrix containing the elements are Pac-man, walls, dots, capsules, ghosts, and scared ghosts. It is helpful to add a new dimension to the state in which the locations of elements are represented by independent matrices. In each matrix, a 0 or 1 respectively express the existence or absence of the element on its corresponding matrix. The obtained reward by the Reinforcement learning agent corresponds with the difference in the game score. In this set-up, Pac-man is get a positive reward for eating dots and that Pacman will get the highest points when it is eat all dots in that grid or maze. A loss (eaten by ghosts) results in a high negative reward. A small negative reward is given over time, in order to advocate quick solutions.

IV.EXPERIMENTAL RESULTS

After Training thousand of episodes, Pacman will play well in the small and medium-sized grids in the game. However, there is a very huge difference between the behavior of a trained model at the beginning of the game and the behavior in the end game. The ghosts are easily caught in the Pacman at the beginning of the training. After completing thousand of the training episodes the Pacman will play effectively. Strikingly, the lack of performance in the end-game also applies to tasks occurring both at the beginning of the game and in the end-game, such as avoiding collisions with ghosts. To address the problem of the decline in performance throughout a game, three hypotheses have been tested to understand this (depressed) behavior. Important to note is that the main characteristic of an end-game scenario is that the amount of dots in the map is sparse. Firstly the problem could be a result of the reward function. Secondly, the sparsity in the matrix could be an issue. And/or lastly, the relatively small amount of observed end-game states could have a negative impact on end-game performance.

The model may train the perfectly Pacman for that Pacman does not easily catch up that ghosts in the grid. The model also trains Pacman on how to escape from the ghosts. And if ghosts are caught that Pacman will lose the game. so every time it has to move away the ghost and eat the dots. The Pacman has to be learned every time it will get a reward from that environment. Firstly every time they play the Pacman gets the reward which is the score. We have to store that training model every time when that Pacman was trained. And continue to the training that points only not from the starting every time.

V.CONCLUSION

This study evaluates the effectiveness of Deep Q-learning and Reinforcement Learning applied to Pac-man. The paper includes background theory of Reinforcement Learning techniques and describes how these techniques were combined to create a model capable of learning to play Pac-man. Experiments were carried out to evaluate the strengths and weaknesses of the used algorithm. The model wins 95% of played games on a medium map (Section 2.2.1). Therefore, the deep q-learning algorithm was successfully play that pacman. A further study could includes the performance of the model on larger maps, including the map size of the original game. An additional weakness is the fact that, for now, the states are represented in a general but hand-made manner. Ultimately the observations carried out by the agent could be, the even more general, pixel values of the game screen. Considerably more work will need to be done to solve the (yet unsolved) problem of playing Pac-man on the original map size and using pixel values of the game screen.

References

[1] S. Adam, L. Bu¸soniu, R. Babuˇska, Systems Man and Cybernetics Part C: Applications and Reviews IEEE Transactions on 2012, 42, 201–212.
[2] M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, Journal of Artificial Intelligence Research 2012. [3] R. Bellman, Indiana Univ. Math. J. 1957, 6, 679–684.
[4] R. E. Bellman, S. E. Dreyfus, Applied dynamic programming, Princeton university press, 2015.
[5] Berkeley Pacman project, http://ai.berkeley.edu/project_overview.html.
[6] T. J. Davidson, F. Kloosterman, M. A. Wilson, Neuron 2009, 63, 497–507.
[7] J. DeNero, D. Klein in Proceedings of the Symposium on Educational Advances in Artificial Intelligence, 2010.
[8] Google, Tensorflow, https : / / www . tensorflow . org / versions / r0 . 8 / get _ started / index.html, 2016.
[9] D. H. Hubel, T. N. Wiesel, The Journal of physiology 1968, 195, 215–243.
[10] A. Krizhevsky, I. Sutskever, G. E. Hinton in Advances in neural information processing systems, 2012, pp. 1097–1105