2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE) | 978-1-7281-9146-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICAICE51518.2020.00033

Analysis of Artificial Intelligence Applied in Video Games

Chengshuo Jiang
Johns Creek High School
Johns Creek, GA 30022, USA
dannycn2002@gmail.com

***Abstract*—Deep reinforcement learning is growing faster than ever before in the artificial intelligence world. As its applications and accomplishments proliferate, the challenges it faces will also increase in number. In this paper, the author focuses on deep reinforcement learning being applied to the field of games and video games. S some of the achievements in various types of games, from 2D perfect information environment (board games) to 3D imperfect information environment (first person perspective games) and the various challenges that DRL faces when being applied to the field will be analyzed. Challenges such as sample efficiency, exploration & exploitation trade-off, and delayed & sparse rewards will be discussed. In addition, some solutions are suggested. The solutions might also be applied to the various game examples in order to improve their performance.**

Keywords-deep reinforcement learning; Game AI; videogames; board games; sample efficiency; delayed & sparse rewards

I. INTRODUCTION

As the need for artificial intelligence grows in our world today, the variety of applications of AI also grows. Some of its applications include object recognition, medical, translation, speech recognition, personalization, and others. Among them, AI in video games is a well-established research area. It studies the interaction between the game environment and the game agents, and how such AIs are capable of achieving or sometimes surpassing human level gameplay. While there are many genres of games, each with a different play style, they offer a variety of game environments and tasks for the agents. Video games also provide an infinite supply of fully labelled data, controllable and replicable environment, and less safety and ethical concerns [1]. One of the most active areas in artificial intelligence research today is reinforcement learning, which is a technique in which an agent tries to achieve as much 'reward' for its actions as possible in an unknown environment. Conversely, deep learning is another technique in which a program or neural network trains using training datasets to perform a specific task, and after its training it is used to perform that task on its own. By combining the two techniques, deep reinforcement learning can be obtained, which is actually reinforcement learning but with deep learning neural networks. This paper will go over the approaches and structures of deep reinforcement learning in several genres of video games, such as board games, Atari 2600 games, and first-person-perspective games. The author will also discuss some key points when applying reinforcement learning to video games, including exploration and exploitation, sample efficiency, and delayed spare rewards.

II. ANALYSIS ON BOARD GAMES

A. AlphaGo

Perhaps one of the most notable examples of deep reinforcement learning used in board games is Google Deepmind's AlphaGo and AlphaGo Zero. Go is an ancient Chinese board game that utilizes a 19x19 board, where players use their stones (white or black) to surround opposing stones on all orthogonally adjacent points. Due to its astronomically large state space, or possible moves, Go was considered to be extremely challenging for an AI to win [2]. But by using the Monte Carlo Tree Search, or MCTS, AlphaGo is able to find the optimal actions much more efficiently than previously known search methods such as minimax. AlphaGo Zero is a more recent generation in the AlphaGo series. Unlike its predecessors, however, AlphaGo Zero uses self-play reinforcement learning, meaning that it played against itself. Previous AlphaGo trained AIs used supervised learning, which turned out to limit their full potential. Since they are trained with games that are played by humans, they can only get as good as humans [2]. AlphaGo Zero also utilizes a singular combined residual neural network for its move probability distribution p and the probability of winning in the current state v, whereas AlphaGo uses two separate convolutional networks. Due to the instability of pure self-play reinforcement learning, MCTS is used to compute another set of move probability distribution π and probability of winning z. Next, p is compared to π and v is compared to z, and loss is calculated using the following equation,

$$l = (z - v)^2 - \pi^T \log p + c\|\theta\|^2 \quad (1)$$

where θ are the parameters of the residual neural network, and c is a parameter used to prevent over-fitting. Creating an AI that can win Go was considered extremely challenging, but AlphaGo and AlphaGo Zero were able to overcome this hurdle and become the world's top Go players [2].

B. Deep Blue

Another example of deep reinforcement learning being used in board games is Deep Blue, which was able to beat the chess world champion in 1997. Due to the technology in 1997, being not as advanced as it is in the 2010s, more Deep Blue contains more 'brute force' code than AlphaGo, using minimax instead of MCTS. Deep Blue has both software and hardware components. The hardware consists of chips, which can be split into three components, the evaluation function, alpha-beta search control, and new move generator [3]. The evaluation

978-1-7281-9146-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICAICE51518.2020.00033

function can be separated into three more parts, which are the piece placement evaluation, endgame evaluation and slow evaluation. The piece placement evaluation assigns a value for each piece on every square of the chess board and changes those values when one piece captures another [3]. The endgame evaluation keeps the counts of various chess pieces still on the board, which are then used to compute endgame assertions such as rule draws and insufficient materials. The slow evaluation computes values for various chess concepts, which include square control, pins, king safety etc. The alpha-beta search control contains a 16-bit data path controlled by three state machines, two of which indirectly controls the move generator [3]. The move generator is fairly self-explanatory; it generates the moves that follow the game rules based on the state. Since the 1997 victory, chess-playing computer programs have built upon Deep Blue's technology to become even more proficient and efficient [3].

III. Analysis on Atari 2600 Games—MuZero

Another creation from Google's Deepmind Lab is the MuZero AI, and it could be considered a late successor of AlphaGo Zero. After AlphaGo Zero, AlphaZero was created and could play Go. Shogi and Chess. MuZero stemmed from AlphaZero and could play a variety of Atari games as well as board games [4]. MuZero functions similarly to AlphaGo Zero, where MCTS is used to compute π (search policy), z (search value), and u (observed reward). Next, they are then compared to their respective parts predicted by the network, which are p (predicted policy), v (predicted value), and r (predicted reward). One thing to keep in mind is that instead of using the terminal node value provided by the simulator for v (used by AlphaZero and AlphaGo Zero), MuZero treats terminal nodes indifferently and always uses the value predicted by the network. The comparison is done through the use of loss functions l^p, l^v, and l^r respective to policy, value and reward. The total loss is computed through the following general equation:

$$l_t(\theta) = \sum_{k=0}^{K} l^r(u_{t+k}, r_t^k) + l^v(z_{t+k}, v_t^k) + l^p(\pi_{t+k}, p_t^k) + c||\theta||^2 \tag{2}$$

where θ are the parameters of the model, and c is a parameter used to prevent over fitting [4].

TABLE I. Evaluation of MuZero in Atari for individual games with 30 random no-op starts [4]

Game	Random	Human	SimPLe [20]	Ape-X [18]	R2D2 [21]	*MuZero*	*MuZero* normalized
alien	227.75	7,127.80	616.90	40,805.00	229,496.90	**741,812.63**	10,747.5 %
amidar	5.77	1,719.53	74.30	8,659.00	**29,321.40**	28,634.39	1,670.5 %
assault	222.39	742.00	527.20	24,559.00	108,197.00	**143,972.03**	27,664.9 %
asterix	210.00	8,503.33	1,128.30	313,305.00	**999,153.30**	998,425.00	12,036.4 %
asteroids	719.10	47,388.67	793.60	155,495.00	357,867.70	**678,558.64**	1,452.4 %
atlantis	12,850.00	29,028.13	20,992.50	944,498.00	1,620,764.00	**1,674,767.20**	10,272.6 %
bank heist	14.20	753.13	34.20	1,716.00	**24,235.90**	1,278.98	171.2 %
battle zone	2,360.00	37,187.50	4,031.20	98,895.00	751,880.00	**848,623.00**	2,429.9 %
beam rider	363.88	16,926.53	621.60	63,305.00	188,257.40	**454,993.53**	2,744.9 %
berzerk	123.65	2,630.42	-	57,197.00	53,318.70	**85,932.60**	3,423.1 %
bowling	23.11	160.73	30.00	18.00	219.50	**260.13**	172.2 %
boxing	0.05	12.06	7.80	**100.00**	98.50	**100.00**	832.2 %
breakout	1.72	30.47	16.40	801.00	837.70	**864.00**	2,999.2 %
centipede	2,090.87	12,017.04	-	12,974.00	599,140.30	**1,159,049.27**	11,655.6 %
chopper command	811.00	7,387.80	979.40	721,851.00	986,652.00	**991,039.70**	15,056.4 %
crazy climber	10,780.50	35,829.41	62,583.60	320,426.00	366,690.70	**458,315.40**	1,786.6 %
defender	2,874.50	18,688.89	-	411,944.00	665,792.00	**839,642.95**	5,291.2 %
demon attack	152.07	1,971.00	208.10	133,086.00	140,002.30	**143,964.26**	7,906.4 %
double dunk	-18.55	-16.40	-	**24.00**	23.70	23.94	1,976.3 %
enduro	0.00	860.53	-	2,177.00	2,372.70	**2,382.44**	276.9 %
fishing derby	-91.71	-38.80	-90.70	44.00	85.80	**91.16**	345.6 %
freeway	0.01	29.60	16.70	**34.00**	32.50	33.03	111.6 %
frostbite	65.20	4,334.67	236.90	9,329.00	315,456.40	**631,378.53**	14,786.7 %
gopher	257.60	2,412.50	596.80	120,501.00	124,776.30	**130,345.58**	6,036.8 %
gravitar	173.00	3,351.43	173.40	1,599.00	**15,680.70**	6,682.70	204.8 %
hero	1,026.97	30,826.38	2,656.60	31,656.00	39,537.10	**49,244.11**	161.8 %
ice hockey	-11.15	0.88	-11.60	33.00	**79.30**	67.04	650.0 %
jamesbond	29.00	302.80	100.50	21,323.00	25,354.00	**41,063.25**	14,986.9 %
kangaroo	52.00	3,035.00	51.20	1,416.00	14,130.70	**16,763.60**	560.2 %
krull	1,598.05	2,665.53	2,204.80	11,741.00	218,448.10	**269,358.27**	25,083.4 %
kung fu master	258.50	22,736.25	14,862.50	97,830.00	**233,413.30**	204,824.00	910.1 %
montezuma revenge	0.00	**4,753.33**	-	2,500.00	2,061.30	0.00	0.0 %
ms pacman	307.30	6,951.60	1,480.00	11,255.00	42,281.70	**243,401.10**	3,658.7 %
name this game	2,292.35	8,049.00	2,420.70	25,783.00	58,182.70	**157,177.85**	2,690.5 %
phoenix	761.40	7,242.60	-	224,491.00	864,020.00	**955,137.84**	14,725.3 %
pitfall	-229.44	**6,463.69**	-	-1.00	0.00	0.00	3.4 %
pong	-20.71	14.59	12.80	**21.00**	**21.00**	**21.00**	118.2 %
private eye	24.94	**69,571.27**	35.00	50.00	5,322.70	15,299.98	22.0 %
qbert	163.88	13,455.00	1,288.80	302,391.00	**408,850.00**	72,276.00	542.6 %
riverraid	1,338.50	17,118.00	1,957.80	63,864.00	45,632.10	**323,417.18**	2,041.1 %
road runner	11.50	7,845.00	5,640.60	222,235.00	599,246.70	**613,411.80**	7,830.5 %
robotank	2.16	11.94	-	74.00	100.40	**131.13**	1,318.7 %
seaquest	68.40	42,054.71	683.30	392,952.00	**999,996.70**	999,976.52	2,381.5 %
skiing	-17,098.09	**-4,336.93**	-	-10,790.00	-30,021.70	-29,968.36	-100.9 %
solaris	1,236.30	**12,326.67**	-	2,893.00	3,787.20	56.62	-10.6 %
space invaders	148.03	1,668.67	-	54,681.00	43,223.40	**74,335.30**	4,878.7 %
star gunner	664.00	10,250.00	-	434,343.00	**717,344.00**	549,271.70	5,723.0 %
surround	-9.99	6.53	-	7.00	9.90	**9.99**	120.9 %
tennis	-23.84	-8.27	-	**24.00**	-0.10	0.00	153.1 %
time pilot	3,568.00	5,229.10	-	87,085.00	445,377.30	**476,763.90**	28,486.9 %
tutankham	11.43	167.59	-	273.00	395.30	**491.48**	307.4 %
up n down	533.40	11,693.23	3,350.30	401,884.00	589,226.90	**715,545.61**	6,407.0 %
venture	0.00	1,187.50	-	1,813.00	**1,970.70**	0.40	0.0 %
video pinball	0.00	17,667.90	-	565,163.00	**999,383.20**	981,791.88	5,556.9 %
wizard of wor	563.50	4,756.52	-	46,204.00	144,362.70	**197,126.00**	4,687.9 %
yars revenge	3,092.91	54,576.93	5,664.30	148,595.00	**995,048.40**	553,311.46	1,068.7 %
zaxxon	32.50	9,173.30	-	42,286.00	224,910.70	**725,853.90**	7,940.5 %
# best	0	5	0	5	13	37	

MuZero achieved a new state of the art for both mean and median normalized scores in 57 games within the ALE (Arcade Learning Environment, provides the environment for a variety of Atari games), surpassing not only humans but also previous state of the art approaches [4].

IV. Analysis on First-person Perspective Games—Doom

Unlike board games and Atari games, first-person perspective games only offer the games their own 'first-person' perspective, resulting in imperfect information input; however, tasks that involve first-person perspective games are suitable for real-world robotics application [1]. Using an engine called ViziDoom to simulate the environments of the first-person shooting game Doom, an unnamed model is tasked with playing deathmatch games, which involves navigating through the map and shooting enemies when observed. The agent is assigned two different phases, which include the navigation phase, where the agent explores the map, and action phase, where the agent fights observed enemies; depending on whether there is an enemy in the current frame, the agent will switch between the phases [5]. To counteract the imperfect information input, the model is built on top of a Deep Recurrent Q-Network. Unlike Deep Q-Networks, where models estimate $Q(s_t, a_t)$, this model estimates $Q(h_{t-1}, o_t, a_t)$ or $Q(h_t, a_t)$, where o is the imperfect information that the agent observes, h is an input that represents the hidden state of the agent, and $h_t = LSTM(h_{t-1}, o_t)$. Instead of simply giving positive reward when getting kills and negative reward when suiciding, the model implements intermediate rewards such as positive rewards for picking up objects (health, weapons, ammo) and negative rewards for actions such as losing health or wasting ammo for the action phase. Since the navigation phase has a separate network, a positive reward is given for object pickup and a negative reward is given for health loss during this phase [5]. The agent is then trained and has its performance compared to humans in both single player (Agent vs Bots compared to Human vs Bots) and multiplayer (Agent vs Human). The results are displayed in table 2, where we can see that the agent outperformed humans in both modes.

TABLE II. Comparison of human players with agent

	Single Player		Multiplayer	
Evaluation Metric	Human	Agent	Human	Agent
Number of objects	5.2	9.2	6.1	10.5
Number of kills	12.6	27.6	5.5	8.0
Number of deaths	8.3	5.0	11.2	6.0
Number of suicides	3.6	2.0	3.2	0.5
K/D Ratio	1.52	5.12	0.49	1.33

*Single player is both humans and the agent playing against bots in separate games. Multiplayer scenario is agent and human playing against each other in the same game [5].

V. Discussion

A. Sample efficiency

While deep reinforcement learning algorithms have the potential to outclass human performance in a multitude of tasks, they require a lot more training and samples than humans do in order to reach the same performance. Whereas humans can quickly acquire rewarding actions in an environment, it would take deep reinforcement learning algorithms millions of samples to train on and reach a similar level of humans [1]. An effective method that helps deal with sample inefficiency in deep learning is by focusing on learning the best model instead of learning the best policy, and adding a mechanism that stops gathering samples in collection phases when the incoming data resembles previously acquired data. By doing so, the framework is able to avoid repetitive data and efficiently process data[6]. The results of these implementations can be seen through Ready Policy One, a framework developed by researchers from Google, the University of Oxford, and UC Berkeley. The framework was tested on various continuous control tasks from the OpenAI Gym, and had its performance compared to other frameworks at a certain timestep. Ready Policy One outperformed both the greedy approach and the variance + reward approach [7].

B. Exploration-Exploitation

The tradeoff between exploration, the process of gathering more information outside of the current known space, and exploitation, the process of gathering and using information only in the current known space, has always been a challenge for reinforcement learning [1]. Some methods for addressing this trade-off include action space noise, parameter space noise and curiosity driven exploration [8]. Action space noise and parameter space noise are similar in nature, where there is a chance of doing something random (noise) every time an action is taken or the parameters are updated, but parameter space noise has been shown to be more efficient than action space parameter. Curiosity driven exploration, on the other hand, is not based on blind random searches, but based on a 'rewarding curiosity'. The older versions of such exploration methods record the amount, and a positive reward is given when exploring states that have a low number of or no visits. These methods help exploration during the training process by making it explore efficiently while also maintaining exploitation [8].

C. Delayed & Sparse Reward

Delayed & sparse rewards are something that appear quite often in deep reinforcement learning, especially in video games [1]. A lot of times, the rewards are spread out and require exploration in order to be acquired, but inefficient exploration methods can often lead to sample inefficiency. Oftentimes the

agent has to reach the final goal in order to receive reward. But since only the final goal contains the highly positive reward, the agent is not receiving immediate feedback (thus delayed) on whether or not it is on the right track, which may require more exploration and thus less sample efficiency. A straightforward method for treating delayed rewards and sparse rewards is by simply utilizing more efficient exploration and exploitation algorithms [9]. An example is curiosity driven exploration algorithms, recent curiosity driven exploration algorithms work differently than the ones explained earlier, instead of looking at the number of recorded visits, the algorithm tries to predict the next state given an action, and compares the new state with the predicted state, the larger the error, the higher the intrinsic reward the agent receives [10]. In turn, the agent will receive a high intrinsic reward when exploring unfamiliar states (since the error will likely be high) and low intrinsic reward when exploring familiar states (since the error will most likely be low) [10]. With efficient exploration algorithms, an agent is able to more efficiently use samples to find the optimal policy and thus actions, despite the rewards being delayed or sparse.

These issues described above play a major role in deep reinforcement learning development up until today. All of the examples of deep reinforcement learning being applied to games had to deal with one or more of the issues, with each example using a different way of solving the problems. For example, the first-person perspective game example had to deal with sparse rewards. In order to solve this problem, the agent was set to receive a reward for moving away from its previous spots, this helped the agent avoid staying in the same spot or spinning in circles by acting as an artificial curiosity driven exploration. AlphaGo had to play $5.4*10^9$ games for training [11], while it may have been counted as efficient in its times, more recent methods may help it achieve a similar level of gameplay using less training games, a suggestion is using a similar framework to the Ready Policy One framework.

VI. Conclusion

Nowadays, the fast-growing field of AI is showing great promises, developments in deep reinforcement AI has led to many groundbreaking accomplishments. In this paper, some of these accomplishments in the field of games and video games have been analyzed, such as AlphaGo and AlphaGo Zero in Go, MuZero in Atari games, and an unnamed AI in the first-person shooter game Doom, as well as the methods used for them to function the way they are. The author also discusses some of the present challenges that deep reinforcement learning when being applied to the games and video games field such as sample efficiency, exploration and exploitation trade-off, and delayed & sparse rewards, and provides some simple suggestions that help address these challenges. However, the challenges presented above have not been explored completely and fully, thus is open to further future studies. This paper does not go very in depth on the examples provided and is not meant to be relied on for in depth analysis, but instead for an overview of deep reinforcement learning's presence in video games and games. It is hoped that AI as well as deep reinforcement learning will make breakthroughs and have much more influence on the video games, even on the other fields.

Acknowledgement

I would like to extend my thanks to Professor Lio and teaching assistant Jimmy Zhao for sparking my interest as well as teaching me the basic principles of the deep reinforcement learning field.

References

[1] K. Shao, et al. "A Survey of Deep Reinforcement Learning in Video Games." ArXiv:1912.10944 [Cs], Dec. 2019. arXiv.org, http://arxiv.org/abs/1912.10944.

[2] D. Silver,et al. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." ArXiv:1712.01815 [Cs], Dec. 2017. arXiv.org, http://arxiv.org/abs/1712.01815.

[3] M. Campbell, et al. "Deep Blue", Jan. 2002 https://core.ac.uk/download/pdf/82416379.pdf

[4] J. Schrittwieser, et al. "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model." ArXiv:1911.08265 [Cs, Stat], Feb. 2020. arXiv.org, http://arxiv.org/abs/1911.08265.

[5] Guillaume, Lample, and Devendra Singh Chaplot. "Playing FPS Games with Deep Reinforcement Learning." ArXiv:1609.05521 [Cs], Jan. 2018. arXiv.org, http://arxiv.org/abs/1609.05521.

[6] "Researchers Develop Technique to Increase Sample Efficiency in Reinforcement Learning." VentureBeat, 11 Feb. 2020, https://venturebeat.com/2020/02/11/researchers-develop-technique-to-increase-sample-efficiency-in-reinforcement-learning/.

[7] P. Ball, et al. "Ready Policy One: World Building Through Active Learning." ArXiv:2002.02693 [Cs, Stat], Feb. 2020. arXiv.org, http://arxiv.org/abs/2002.02693.

[8] Y. Yang. "Towards Sample Efficient Reinforcement Learning." IJCAI (2018).

[9] J. Hare, "Dealing with Sparse Rewards in Reinforcement Learning." ArXiv:1910.09281 [Cs, Stat], Nov. 2019. arXiv.org, http://arxiv.org/abs/1910.09281.

[10] D. Pathak, et al. "Curiosity-Driven Exploration by Self-Supervised Prediction." ArXiv:1705.05363 [Cs, Stat], May 2017. arXiv.org, http://arxiv.org/abs/1705.05363.

[11] Sample Efficient Reinforcement Learning. https://research.zalando.com/welcome/mission/research-projects/sample-efficient-reinforcement-learning/ [Accessed on 15 Aug. 2020].