Deep Reinforcement Learning of an Agent in a Modern 3D Video Game

Samuel Arzt, Gerhard Mitterlechner, Markus Tatzgern, and Thomas Stütz

Salzburg University of Applied Sciences, Urstein Süd 1, A5412 Puch/Salzburg
thomas.stuetz@fh-salzburg.ac.at

Abstract. This paper applies two recent deep reinforcement learning (DRL) methods, namely *DQN* and *A2C*, to train an agent in a modern 3D video-game environment called *Delivery Duel*. The agent perceives the environment only by raw pixel data. The reward of the DRL algorithms is based on the game's score. The focus of this work is on the impact of environment modifications on the learning performance, namely the environment representation (complex 3D, simple 2D), the reward signal (sparse, continuous) and the agent's motion control (physics-based, linear). The introduced test framework will be made publicly available and thus can be used for future experiments.

Keywords: Deep learning, Reinforcement Learning, video game, 3D

1 Introduction

Recent advances in deep learning have led to major improvements in computer vision, in particular for image classification and object detection tasks (e.g.,[5, 6, 12, 24]). These advances were mainly achieved by supervised learning of convolutional neural networks (CNNs) [14], but have also sparked new interest in revisiting these methods for the field of reinforcement learning (RL). In RL an agent learns through interaction with an environment and receiving feedback in the form of a numerical reward signal [23]. While it was previously thought [16] that the combination of simple online RL and non-linear function approximators, such as deep Neural Networks (NNs), was fundamentally unstable, Mnih et al. [16] showed that the inherent problems of this combination can be overcome by their *DQN* (Deep Q-Network) architecture. Mnih et al. [16] successfully applied their approach to a collection of Atari2600 games, which have been specifically compiled to evaluate machine learning approaches [2]. Further improvements on the Atari benchmark were achieved with improved RL approaches, such as *A2C* (Advantage Actor Critic) [15] and extensions to DQN [10].

Even though these machine learning techniques are tested and compared on simulated game environments, the ultimate goal is to explore new possibilities to solve real-world problems. In this work, two modern deep RL algorithms, namely DQN [16] and A2C [15], are applied to a novel modern 3D video-game called *Delivery Duel*.

The video-games evluated as part of the Atari benchmark are visually simple and resemble the simple rendering variant of the game environment introduced in this paper (see Fig. 1, right). The main question is whether deep RL with DQN [16] and A2C [15] can also be successfully applied to the visually more complex 3D renderings of a modern computer game (see Fig. 1, left). Additionally, the impact of the complexity of the agent's motion control in the environment and the design of the reward function on learning performance is investigated. The novel evaluation framework is publicly accessible online [1].

2 Related Work

RL has been applied to traditional board games such as Backgammon [25] or Go [21, 22], but also card games such as Poker [9, 4] for many years. Video games became a popular testbed for RL algorithms since the introduction of the Arcade Learning Environment (ALE), an API for using the raw pixel data of a collection of Atari2600 games as the input of RL algorithms, by Bellemare et al. [2]. Mnih et al. [17] successfully tackled the task of learning multiple Atari games using the same hyperparameters and network architecture with their DQN algorithm. Multiple extensions of DQN, such as Prioritized Experience Replay [20], Dueling DQN [30], Double DQN [29], Noisy Nets [19, 8] and finally an accumulation of all of these extensions dubbed Rainbow [10] have been proposed. Improvements on the ALE benchmark were also achieved by other algorithms, such as A3C / A2C [15] and ACKTR [31].

Efforts have also been made to apply RL to 3D video-games. Similar to the ALE for arcade games, an API for interacting with the first person shooter (FPS) *Doom* (originally released in 1993) and retrieving its visual output was established by Kempka et al. [11]. The Doom test-environment has since been used in multiple studies, e.g., [13, 31]. In FPSs the game state is usually only partially observable, thus agents have to utilize additional complex extensions such as recurrent network structures, Curriculum Learning and Attention Frames [31], or dividing the problem into a navigation and action phase [13].

Although compared to Atari games, Doom is a more modern game, which uses simple 3D renderings, these studies still very much focus on relatively simple graphics (i.e., low visual complexity, simple textures, little detail). The environment introduced in this paper relies on modern 3D graphics of the Unity [28] game-engine and applies state-of-the-art deep RL algorithms. Instead of complex extensions to these RL algorithms, the focus is on the impact of different environment representations, the motion-control in the environment and the design of the reward function.

3 Deep Reinforcement Learning of an Agent in the 3D Game "Delivery Duel"

Two recent deep RL methods, namely DQN [16, 30, 29, 20] and A2C [15] were applied to a modern 3D video game called Delivery Duel. The game environment

and mechanics are explained in section 3.1. The applied learning approaches and the employed software frameworks are briefly described in section 3.2. Finally, the different configurations of the environment are explained (see section 3.3). All material to reproduce the results of this paper are available online [1].

3.1 3D Game Environment: Delivery Duel

In Delivery Duel the player / agent controls a delivery van. The agent's goal is to deliver pizzas to target locations in an urban environment. The entire scene is rendered from an approximately 75° top-down perspective (see Fig. 1, left). Delivery destinations are highlighted above the corresponding building by a big circular marker. The agent completes a delivery by conducting a throw action in the correct direction and hitting the destination building. After a delivery, the agent has to return to its base, located in the middle of the city, before a new destination is highlighted. The agent's score is increased for delivering items, proportional to how fast the item was delivered, and for returning to the base. The game play builds upon the physics-based motion control, which makes steering the delivery van quite difficult but fun.

3.2 Frameworks: OpenAI Gym / Baselines and Unity ML-Agents

Delivery Duel was developed with the Unity game-engine [28], a very widely employed engine. In order to train RL agents in this environment, the frameworks *Gym* [3], *Baselines* [7] and *ML-Agents* [26] were combined. Baselines provides open-source Python implementations of popular RL algorithms [7]. Gym [18] defines an open-source Python interface for agent environments. ML-Agents extends the Unity engine with an interface for testing AI algorithms on existing Unity games by offering an API which connects Unity with Python [26, 27]. ML-Agents was integrated into Delivery Duel and its Python interface was wrapped as a Gym environment. In this way the DQN and A2C implementation of Baselines was used to train RL agents on Delivery Duel.

The DQN implementation used is a combination of Prioritized Experience Replay [20], Double DQN [29] and Dueling DQN [30], and uses ϵ-greedy exploration. The network structure and all hyperparameters of DQN were set according to [16, 30, 29, 20]. The used CNN consists of three convolutional layers which lead into another fully connected hidden layer. The hyperparameters of A2C were set according to [15]. The input of the network was a stack of the last four frames of raw pixel values, which are gray-scaled and down-sampled to an 84x84 array of 8-bit values, as in [16].

3.3 Evaluated Configurations

In the following, the variation of environment representation, motion control and the reward signal, which result in eight different configurations, are explained. The impact of each of the eight configurations on learning performance of DQN and A2C was evaluated.

Environment Representation: A complex 3D render-mode and a simplified 2D render-mode were evaluated. Fig. 1 shows a visual comparison of these two render-modes.

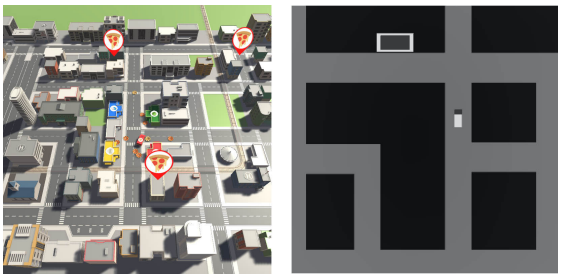

Fig. 1. Comparing the two different render-modes. The city in complex render-mode (*left*), i.e. the same as in the original game, and the city in simple render-mode (*right*), where only the most important information is rendered from a completely top-down view

Agent Motion Control: The agent, which is represented by a delivery van, is allowed to move freely in the environment. The default option of motion control of the agent is physics-based, i.e. the agent moves the van by applying forces to it. The second evaluated motion control approach is a linear motion control model, which allows the agent to directly translate the van in x and y directions.

Reward: The default option for the agent's reward signal is simply the game's score, which is increased for a successful delivery (by a value in range $[50, 150]$, depending on the speed of delivery) and for returning to the base (by a fixed value of 75). Additionally, a second option with a more continuous reward signal, which also increases or decreases the score by a value of 10 for every five in-game units driven closer or further away from the current target was evaluated.

4 Results

DQN and A2C were used to train an agent for five million time-steps on Delivery Duel using eight different test-configurations (see Sec. 3.3). During training, a single episode was defined as the steps required by the agent to reach a score larger than 1500, at which point the task is considered solved. Furthermore, an episode was terminated if the agent failed to reach its current target for more than 750 steps. The performance of a test-run is measured by the mean reward of the last 100 episodes (mean100). Fig. 2 and 3 compare the performance of agents trained using DQN and A2C on five of the eight different configurations. The remaining three configurations were omitted, because both algorithms were

not able to exceed a mean100 reward of 250. The axes of these figures plot the elapsed time-steps (on the x-axis, from left to right) against the current mean100 reward (on the y-axis, from bottom to top).

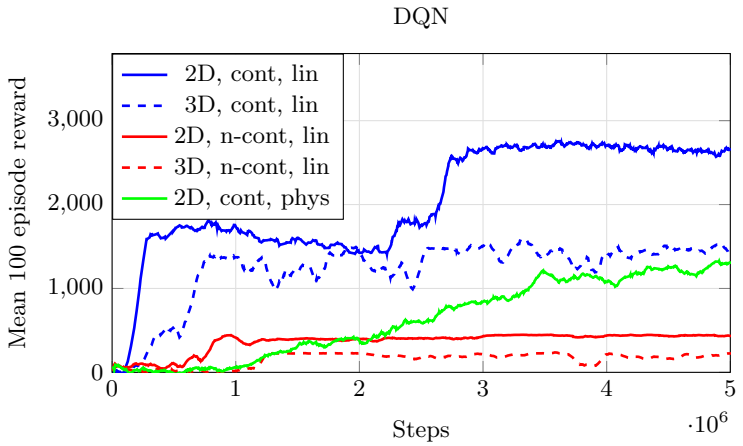

Fig. 2. A comparison of the performance of each DQN agent trained on a different configuration of the environment. The three possible modifications were abbreviated, with 2D / 3D = simple / complex render-mode; (n-)cont = (non-)continuous reward signal; lin / phys = linear / physical motion control

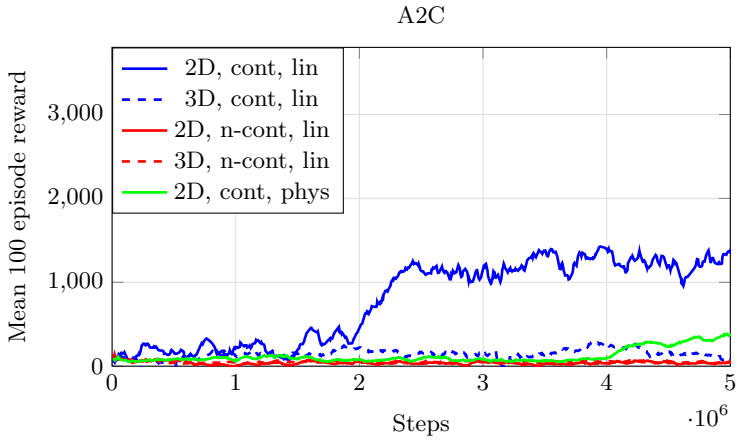

Fig. 3. A comparison of the performance of each A2C agent trained on a different configuration of the environment. The three possible modifications were abbreviated in the same way as for Fig. 2

Overall, DQN performed better than A2C by a large margin after the same amount of steps. However, due to the parallelization capabilities of A2C, it is

able to achieve significantly more steps per second than DQN (approximately by a factor of six).

The configurations using a continuous reward signal and linear movement outperformed all other test configurations. The slightly more continuous reward function is most beneficial for learning performance compared to other variations. Neither DQN nor A2C were able to achieve a mean100 reward larger than 250 when using the unmodified reward signal.

5 Discussion and Outlook

There were multiple reasons for choosing Delivery Duel as an environment. Firstly, unrestricted source-code access allowed to modify the game accordingly in order to compare how subtle changes in a test environment influence training performance. Secondly, Delivery Duel's sparse reward signal poses a significant challenge for end-to-end RL. Thirdly, establishing an interface to the Unity engine for state-of-the-art RL algorithms enables these algorithms to be potentially tested and compared on thousands of other Unity environments.

Interestingly, the two modifications of a more continuous reward signal and linear movement had a greater influence on learning than the render-mode, which, to the human observer, might seem like a more significant change of the environment. For the first 2.5 million steps the DQN agent using the complex render-mode, a continuous reward signal and linear movement even achieves almost the same performance as its counterpart using the complex render-mode (compare *2D, cont, lin* and *3D, cont, lin* of Fig. 2).

When the complex rendering was paired with the physics-based motion control, neither DQN nor A2C were able to produce any successful policies. These results indicate that the employed CNN is able to cope with the additional complexity of a 3D representation, only as long as the underlying environment interaction problem (motion control) remains simple enough. Future work should also consider the impact of different network architectures (e.g., deeper networks, recurrent networks).

Early tests have indicated that agents trained using the linear motion control can be used to control an agent using the physics-based motion control with surprising accuracy. These transferred skills could be utilized in a manner similar to curriculum learning in order to be able to either learn even more complex tasks or to shorten the required training time.

In addition to the single-player mode, Delivery Duel also offers a four player local multi-player. Using this mode, further experiments could be conducted on agent interaction or possible transfer learning capabilities from one player to another.

6 Conclusions

A novel modern 3D game environment for RL is presented, which is made publicly available for the research community [1].

Two recent DRL methods, namely DQN and A2C, have been successfully applied to train an agent in this environment. A more continuous reward signal greatly improved the learning performance. DRL agents were able to complete the task even if the input was a complex 3D rendering. In addition to that, they were also able to cope with a complex motion control. However, the applied DRL methods failed to learn successful policies when confronted with a combination of complex 3D rendering and complex motion control.

These results encourage further experiments on this environment, as well as research to combine the two learned capabilities, e.g., by transfer learning or curriculum learning.

7 Acknowledgements

Delivery Duel was developed in collaboration of Katrin-Anna Zibuschka, Lukas Machegger and Samuel Arzt, who approved to make the game publicly available for scientific purposes. Their work and approval is greatly appreciated.

Furthermore we thank the University of Applied Sciences Salzburg for the provided assistance, including scientific advice and research equipment, which has been a great help in conducting this work.

References

[1] Samuel Arzt. *DRL applied to Delivery Duel Repository*. 2018. URL: `https://github.com/ArztSamuel/DRL_DeliveryDuel` (visited on 08/27/2018).

[2] Marc G Bellemare et al. "The Arcade Learning Environment: An evaluation platform for general agents." In: *J. Artif. Intell. Res. (JAIR)* 47 (2013), pp. 253–279.

[3] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: `arXiv:1606.01540`.

[4] Noam Brown and Tuomas Sanholm. "Libratus: The Superhuman AI for No-Limit Poker". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 2017.

[5] Dan C Cires et al. "Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks". In: *Proc. MICCAI*. Vol. 2. 2013, pp. 411–218.

[6] Dan Cires, Ueli Meier, and Jürgen Schmidhuber. "Multi-column Deep Neural Networks for Image Classification". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE* February (2012).

[7] Prafulla Dhariwal et al. *OpenAI Baselines*. `https://github.com/openai/baselines`. 2017.

[8] Meire Fortunato et al. "Noisy networks for exploration". In: *International Conference on Learning Representations (ICLR)*. 2018.

[9] Johannes Heinrich and David Silver. "Smooth UCT Search in Computer Poker." In: *International Conference on Artificial Intelligence (IJCAI)*. 2015, pp. 554–560.

[10] Matteo Hessel et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *arXiv preprint arXiv:1710.02298* (2017). eprint: 1710.02298.

[11] Michał Kempka et al. "Vizdoom: A doom-based ai research platform for visual reinforcement learning". In: *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE. 2016, pp. 1–8.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances In Neural Information Processing Systems* (2012), pp. 1–9.

[13] Guillaume Lample and Devendra Singh Chaplot. "Playing FPS Games with Deep Reinforcement Learning." In: *AAAI*. 2017, pp. 2140–2146.

[14] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE 86, no. 11*. 1998, pp. 2278–2324.

[15] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2016, pp. 1928–1937.

[16] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[17] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[18] OpenAI. *OpenAI Gym*. 2018. URL: http://web.archive.org/web/20180412213846/https://gym.openai.com/ (visited on 04/14/2018).

[19] Matthias Plappert et al. "Parameter Space Noise for Exploration". In: *International Conference on Learning Representations (ICLR)*. 2018.

[20] Tom Schaul et al. "Prioritized Experience Replay". In: *International Conference on Learning Representations (ICLR)* (2016).

[21] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[22] David Silver et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), pp. 354–359.

[23] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. 2nd ed. In progress, Complete draft. MIT Press, 2018, p. 394.

[24] Christian Szegedy et al. "Going Deeper with Convolutions". In: *CVPR*. 2015.

[25] Gerald Tesauro. "Temporal difference learning and TD-Gammon". In: *Communications of the ACM* 38.3 (1995), pp. 58–68.

[26] Unity-Technologies. *ML-Agents Homepage*. 2018. URL: https://web.archive.org/web/20180322013330/https://unity3d.com/machine-learning/ (visited on 04/14/2018).

[27] Unity-Technologies. *ML-Agents Repository*. 2018. URL: https://github.com/Unity-Technologies/ml-agents (visited on 04/14/2018).

[28] Unity-Technologies. *Unity3D*. 2018. URL: http://web.archive.org/web/20180412233455/https://unity3d.com/ (visited on 04/14/2018).

[29] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." In: *AAAI*. 2016, pp. 2094–2100.

[30] Ziyu Wang et al. "Dueling network architectures for deep reinforcement learning". In: *International Conference on Machine Learning (ICML)*. PMLR, 2016, pp. 1995–2003. URL: http://proceedings.mlr.press/v48/wangf16.html.

[31] Yuhuai Wu et al. "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation". In: *Advances in neural information processing systems (NIPS)* (2017), pp. 5285–5294.