Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Arnold: An Autonomous Agent to Play FPS Games

Devendra Singh Chaplot,* **Guillaume Lample***
{chaplot,glample}@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

Abstract

Advances in deep reinforcement learning have allowed autonomous agents to perform well on Atari games, often outperforming humans, using only raw pixels to make their decisions. However, most of these games take place in 2D environments that are fully observable to the agent. In this paper, we present Arnold, a completely autonomous agent to play First-Person Shooter Games using only screen pixel data and demonstrate its effectiveness on Doom, a classical first-person shooter game. Arnold is trained with deep reinforcement learning using a recent Action-Navigation architecture, which uses separate deep neural networks for exploring the map and fighting enemies. Furthermore, it utilizes a lot of techniques such as augmenting high-level game features, reward shaping and sequential updates for efficient training and effective performance. Arnold outperforms average humans as well as in-built game bots on different variations of the deathmatch. It also obtained the highest kill-to-death ratio in both the tracks of the Visual Doom AI Competition and placed second in terms of the number of frags.

1 Introduction

Deep reinforcement learning has proved to be very successful in mastering human-level control policies in a wide variety of tasks. In particular, Deep Q-Networks (DQN) are shown to be effective in playing Atari 2600 games (Mnih et al. 2013) and more recently, in defeating world-class Go players (Silver et al. 2016). The task of playing a First-Person-Shooting (FPS) game in a 3D environment is much more challenging than playing most Atari games as it involves a wide variety of skills, such as navigating through a map, collecting items, recognizing and fighting enemies, etc. Furthermore, states are partially observable, and the agent navigates a 3D environment in a first-person perspective, which makes the task more suitable for real-world robotics applications.

In this paper, we present Arnold, an autonomous agent trained using the architecture introduced by Lample and Chaplot (2017). It divides the problem of playing FPS games into two phases: navigation (exploring the map to collect items and find enemies) and action (fighting the enemies when they are observed), and uses separate Deep Q-networks to learn policies for each phase of the game. The navigation is done by a DQN (Mnih et al. 2013), while the action network is controlled by a DRQN (Hausknecht and Stone 2015). The DRQN network is augmented with high-level game information, such as the presence of enemies on the screen. It is trained to simultaneously learn these features along with minimizing a Q-learning objective, which is shown to dramatically improve the performance.

2 Visual Doom AI Competition

Recently, Visual Doom AI Competition (ViZDoom)[1] was organised to evaluate AI agents playing deathmatchs in Doom. In the deathmatch scenario, all the agents spawn on the same map with the objective to frag as many enemies as possible within a fixed time limit. When an agent dies, it respawns on the same map at another location immediately. The competition consisted of two tracks:

- **Limited deathmatch on a known map:** The map is known beforehand and the only available weapon is a rocket launcher. Agents can gather health packs and ammo, but no other weapon is available.
- **Full deathmatch on unknown maps:** Agents start with a pistol, but can pick up different weapons around the map, as well as gather health packs and ammo. They are evaluated on unknown maps.

Both the tracks consisted of 12 rounds of 10 minutes each, with a maximum of 8 agents by round. The competition organizers also released an API for efficiently interacting with the Doom game engine (Kempka et al. 2016).

3 Training

Arnold was trained on 2 Nvidia GTX Titan X GPUs. It roughly took 3 days to train it on the limited deathmatch, and about 5 days for the full deathmatch, although some minor improvements were still observable after that. It was implemented with UltraDeep[2], a deep learning library built on top

*The authors contributed equally to this work.

[1] ViZDoom Competition at IEEE Computational Intelligence And Games (CIG) Conference, 2016 (http://vizdoom.cs.put.edu.pl/competition-cig-2016)

[2] https://github.com/glample/UltraDeep

of Theano (Theano Development Team 2016). Killing an enemy sometimes can require very long sequences of specific actions (pick up a weapon, aim at an enemy, fire several projectile in its direction) which are extremely unlikely to occur in the initial training phase, where the agent actions are selected randomly. This leads to a very sparse replay memory and makes the training very difficult. We used reward shaping (Ng 2003) in order to help the agent learn favorable actions. We set the frame-skip to 4 (Bellemare et al. 2012) to accelerate the training. A higher frame-skip would make it difficult for the agent to aim at distant enemies. All networks were trained using the RMSProp algorithm and mini-batches of size 32. Network weights were updated every 4 steps, so experiences are sampled on average 8 times during the training (Van Hasselt, Guez, and Silver 2015). The replay memory contained the one million most recent frames. The discount factor was set to $\gamma = 0.99$. We used an ϵ-greedy policy during the training, where ϵ was linearly decreased from 1 to 0.1 over the first million steps, and then fixed to 0.1. We used a screen resolution of 440x225 and resized each frame to 108x60 image before passing it to the model. Although faster, our model obtained a lower performance using grayscale images, so we decided to use colors in all experiments.

4 Results

We use two metrics to compare the performance of all agents participating in the ViZDoom competition[3]: Kill-to-Death (K/D) Ratio and number of frags, i.e. number of bots killed by the agent minus the total number of suicides committed. As shown in Table 1, Arnold outperforms other agents in terms of K/D Ratio by a significant margin. In the full deathmatch, Arnold has a K/D Ratio of 33.40 as compared to a maximum 3.58 among the other agents, while in the limited deathmatch Arnold beats the best among other agents with a K/D Ratio of 2.45 to 1.45. Arnold placed second in both the tracks of the competition as number of frags was used as the metric. Arnold was trained to optimize K/D Ratio. Higher K/D Ratio was likely a result of crouching, which makes Arnold less vulnerable to enemy attacks and it indicates Arnold would be superior in a 1-on-1 deathmatch. However, Arnold movements were significantly slower due to its crouching position, making it harder for it to follow enemies and aim at them, which resulted in a fewer number of frags.

5 Demonstration

At the demonstration, attendees will have the opportunity to play against Arnold in the deathmatch scenario and live games will be shown on a large monitor. We will also have another machine demonstrating the performance of the model (as shown in the supplementary videos) and a poster with technical details. We also plan to demonstrate some intelligent behaviour learned by Arnold such as dodging rockets, avoiding lava and moving backward to avoid suicide when firing too close to a target.

[3]http://vizdoom.cs.put.edu.pl/competition-cig-2016/results

	Limited Deathmatch		Full Deathmatch	
Agent Name	Number of frags	K/D Ratio	Number of frags	K/D Ratio
5vision	142	0.41	12	0.20
AbyssII	118	0.40	-	-
Arnold	413	**2.45**	164	**33.40**
CLYDE	393	0.94	-	-
ColbyMules	131	0.43	18	0.20
F1	**559**	1.45	-	-
IntelAct	-	-	**256**	3.58
Ivomi	-578	0.18	-2	0.09
TUHO	312	0.91	51	0.95
WallDestroyerXxx	-130	0.04	-9	0.01

Table 1: Results of the Visual Doom AI Competition. Scores marked with '-' indicate that the agent did not participate in the corresponding track. The best results in each column are marked in **bold**. Arnold achieves the highest K/D Ratio in both the tracks.

References

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2012. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*.

Hausknecht, M., and Stone, P. 2015. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*.

Kempka, M.; Wydmuch, M.; Runc, G.; Toczek, J.; and Jaśkowski, W. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. *arXiv preprint arXiv:1605.02097*.

Lample, G., and Chaplot, D. S. 2017. Playing fps games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Ng, A. Y. 2003. *Shaping and policy search in reinforcement learning*. Ph.D. Dissertation, University of California, Berkeley.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

Van Hasselt, H.; Guez, A.; and Silver, D. 2015. Deep reinforcement learning with double q-learning. *CoRR, abs/1509.06461*.