Evaluating competition in training of Deep Reinforcement Learning agents in First-Person Shooter games

Paulo Bruno S. Serafim, Yuri Lenon B. Nogueira, Creto A. Vidal, Joaquim B. Cavalcante Neto
Department of Computing
Federal University of Ceará
Fortaleza, Brazil
paulobruno@alu.ufc.br; {yuri, cvidal, joaquimb}@dc.ufc.br

***Abstract*—This work evaluates competition in training of autonomous agents immersed in First-Person Shooter games using Deep Reinforcement Learning. The agents are composed of a Deep Neural Network, which is trained using Deep Q-Learning. The inputs of the networks are only the pixels of the screen, allowing the creation of general players, capable of handling several environments without the need for further modifications. ViZDoom, an Application Programming Interface based on the game Doom, is used as the testbed because of its appropriate features. Fifteen agents were divided into three groups, two of which were trained by competing with each other, and the third was trained by competing against opponents that act randomly. The developed agents were able to learn adequate behaviors to survive in a custom one-on-one scenario. The tests showed that the competitive training of autonomous agents leads to a greater number of wins compared to training against non-intelligent agents.**

***Keywords*-autonomous agents; deep reinforcement learning; digital games; competitive learning; first-person shooter games**

I. Introduction

Competition has an important role in digital action games, especially in First-Person Shooter (FPS) games. In those games, the multiplayer mode is very popular, that called the attention of the game industry, which introduced characters that are controlled by Artificial Intelligence techniques, and that are continuously improved to gain even more autonomy [18], [20], [7]. Ideally, those autonomous agents can learn how to behave in an environment by themselves, as opposed to scripted bots, which behave in accordance with explicitly programmed rules.

However, creating autonomous agents to play FPS games in competitive environments is very hard, because the success of a player is tied to the failure of the opponent. Thus, training autonomous agents is necessary for them to develop adequate behaviors. Although an agent can learn how to play some scenarios in a single-agent manner, using competition may be beneficial to its performance.

The main contribution of this paper is to analyze competitive training of autonomous agents against two different types of opponents, other autonomous agents and non-intelligent (i.e., randomly acting) players, and observe how such agents could benefit from training against these different types. In this work, we are concerned in a prior moment of the development of competitive autonomous agent: training. The novelty presented is performing an experimental evaluation of those types of competitive training and evaluate which type of competitive training can be more effective in relation to the performance of the agents.

Training against non-intelligent opponents can have advantages, like increasing the velocity of the training and execution phases since this type of agent does not need to run its own controller, or ensuring equality in training for any agent because the behavior developed by the agent does not rely on the learning dynamics of its opponents. Our hypothesis is that agents trained by competing against other autonomous agents will perform better than agents which trained against non-intelligent opponents. We conjecture that when agents perform better in competitive tests they will be more suitable to play competitive modes in a FPS game.

In order to test our hypothesis, we use a Deep Reinforcement Learning model, called Deep Q-Networks [26], [27], to control the autonomous agents. The agents are placed in a custom competitive scenario of the API ViZDoom [18], a research platform based on the game Doom (id Software, 1993). After the definition of the controller, the agents are divided into different teams and are tested by competing with each other.

This paper is organized as follows. In Section II, we summarize some works that use Single-Agent and Multi-Agent Deep Reinforcement Learning to play digital games. In Section III, we describe Deep Q-Networks, the main technique used in Deep Reinforcement Learning problems. In Section IV, we present the scenario played by the agents and we describe the settings of the performed experiments. In Section V, we show the obtained results and discuss them. Finally, in Section VI we summarize the work presented, suggesting some future works in Subsection VI-A.

II. Related Work

A. Single-Agent Deep Reinforcement Learning

Mnih *et al.* [26] were the first to use a Deep Learning model in Reinforcement Learning. They trained a Deep

Neural Network using a variant of the Q-Learning algorithm, and, using only raw pixels from the screen as input, they were successful in producing agents that could play digital games. In that pioneering work, a Deep Q-Learning model learned to play seven Atari 2600 games using as input a brute high-dimensional input, only the raw pixel data. The authors also used an adaptation of Q-Learning called Experience Replay [23], which improved the efficiency of data use, increased the learning speed and led to a better choice of parameters to avoid local minima.

Mnih *et al.* [27] developed Actor-Critic model, an improved variant of their previous technique, and tested it in 49 Atari 2600 games. In 29 of the games they reached human-level performance.

Since then, a great number of works used Atari 2600 games as testbeds, such as [29], [22], [25], [31], because the complexity of the games was high, the input of pixels was straightforward, and the games had a high public appeal.

Kempka *et al.* [18] developed ViZDoom as a research platform for Reinforcement Learning problems, which was based on the game Doom (id Software, 1993). That platform allows the development of players controlled by Artificial Intelligence techniques using the screen as their vision. Recently, ViZDoom has been used in several works of Deep Reinforcement Learning [20], [7], [1], [30], [39], [32] as testbed.

Different games were also used as testbeds for Deep Reinforcement Learning [5], [38]. For example, Chen and Yi [5] tested their model in Super Smash Bros and in Mario Tennis; and van Seijen *et al.* [38] tested their technique in Ms. Pac Man.

Those works present the development of a single agent to play digital games. In this paper, we will use multiple agents divided into teams and immersed in a competitive mode of ViZDoom. Thus, we will be able to evaluate multiple autonomous agents in a single environment.

B. Multi-Agent Deep Reinforcement Learning

Recently, Deep Learning models were also used in the area of Multi-Agent Reinforcement Learning (MARL). Approaches that combine Deep Q-Learning with MARL are called Multi-Agent Deep Reinforcement Learning (MADRL), and are divided into three categories: cooperative, competitive and mixed approaches.

Several works in MADRL use digital games as testbeds, in which the agents develop their skills either by means of cooperation with one another [16], [10], [9], [28], [19], [6], [13]; or by means of competition, trying to beat one another [33], [17], [24]; or, yet, by means of cooperation to compete, in which some agents need to cooperate in order to beat other opponents [36], [15], [4], [8], [21].

While competition was already used in some of those works, none of them provided an analysis of how the training can be influenced by the types of agents used. We observed this lack of prior evaluation and present a comparison of competitive training against two different types of opponents, autonomous agents and non-intelligent agents, and how they will affect the performance in a direct competition.

III. Background

A. Deep Q-Network

Reinforcement Learning tasks are sequential decision problems whose objective is to find the policy that maximizes the sum of received rewards. A policy is a set of actions that should be taken by the agent for every possible state. The agent analyzes the current state s, and decides what action, a, to take according to the policy, π. The goal of the agent is to find the policy in which the expected sum of discounted rewards, R_t, is maximum

$$R_t = \sum_{i=t}^{T} \gamma^{\,i-t} r_i \tag{1}$$

where T is the last iteration, t is the current iteration, γ is the discount factor, which varies in the interval [0,1], and r_i is the reward at iteration i. The discount factor determines how fast the future rewards decay with respect to the current reward, which could make immediate rewards more important than the later ones.

The loss function of a parameterized value function Q_θ is given by [20], [26]:

$$L_t(\theta_t) = \mathbb{E}[(y_t - Q_{\theta_t}(s,a))^2 | s, a, r, s'], \tag{2}$$

We compute the gradient of the above loss function using the approximation

$$\nabla_{\theta_t} L_t(\theta_t) \approx (y_t - Q_{\theta_t}(s,a)) \nabla_{\theta_t} Q_{\theta_t}(s,a). \tag{3}$$

The Q-learning updates, using the loss function estimations of (2), are stable and perform well in practice [27].

IV. Experiments

A. Environment

A custom scenario was created to analyze the ability of an agent that is engaged in a gun shooting competition with other players. In that type of FPS game, the winner kills the opponent more times than it gets killed. In this environment, the two competitors (Fig. 1), which are positioned at opposite sides of a rectangular room, can take one out of three possible actions: move left, move right, and shoot.

(a) Green agent. (b) Red agent.

Figure 1. Both agents as seen from opponent's view.

Since this scenario is executed in a client-server connection, some adjustments need to be made to ensure a fair competition. Thus, the environment is executed in a synchronous mode, keeping a fixed frame rate; and, every time an agent gets killed, it is respawned at a random position in its side of the room.

An episode starts with both agents at opposite sides of the scenario (Fig. 2). An agent dies if it is shot once. The goal of an agent is to maximize the number of times it kills the opponent. A secondary implicit goal is to stay alive as long as possible, since dying means that the opponent scored.

1) Evaluation Metrics: The evaluation metrics were chosen to maintain consistency with the scenarios already present in ViZDoom. An agent scores 60 points every time it kills an opponent, and it dies if it is shot once, without scoring any point. When an agent dies, it is respawned in a random location at its side of the room. After 300 steps (see Subsection IV-D for details about a step), the agents receive as reward their accumulated scores and an episode ends.

If an agent performs one action per frame, the difference between the previous states and the current one is so subtle that learning will become hard. To solve this problem, we used a frame repeat value of eight frames. That means an action is chosen, repeated through eight frames and only then another action is chosen [2].

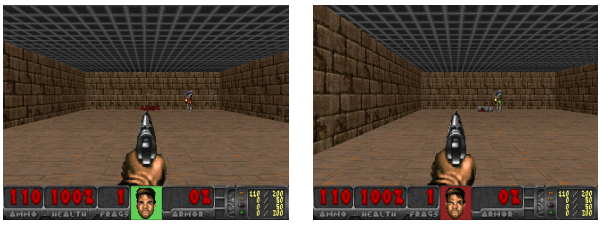
B. Hyperparameters

The following parameters were chosen empirically. Starting with the values presented in [20], the Q-Learning discount factor, γ, was set to 0.99 and the learning rate was equal to 0.0001. An ϵ-greedy policy [35], with linear decay from 1.0 to 0.1, was used to balance the trade-off between exploration and exploitation. The network biases were initialized with a value of 0.1 and all the weights were initialized with random values using Xavier's Initialization [11]. To reduce overfitting, we used the Dropout technique [34] during training with a probability of 0.7.

The basic environment of ViZDoom, as described in [32], was used to evaluate some minor variations. Since the results did not show meaningful variations in performance, the following parameters were chosen to increase training speed. The RMSProp algorithm [37] was used to train the network with mini-batches of size 64. Experience Replay [23] was used to reduce correlation between consecutive frames. A replay memory keeps track of the latest one hundred thousand frames and a randomly chosen sample is passed to the network every update.

C. Neural Network Settings

1) Initial Layers: The input of the neural network is a matrix of real numbers, representing a 64×48 pixels gray scale image, which is then passed to the first convolutional layer (Fig. 3). That layer has a kernel size of four, a stride of size two, which halves the size of the input, and computes 32 features. The outputs are then fed into a second convolutional layer which will compute 64 features. This layer also has a kernel size of four and a stride of size two, halving the size of the input. Both convolutional layers use the ReLU activation function [14], [12].



(a) Green agent point of view. (b) Red agent point of view.

Figure 2. Agents' point of view of the same frame in the environment.

2) Final Layers: After passing through the second convolutional layer, the 64 images of size 16×12 pixels are flattened into arrays of size 192, and are fully connected to a layer of 512 neurons (Fig. 3). This layer has the goal to gather all the features previously discovered in the convolutional layers. The results are sent to the output layer, which has three neurons, one for each possible action that can be taken by an agent.

3) Input and Output: The raw colored 640×480 pixels image of the screen is preprocessed into a smaller one with reduced size and color so that the input of the network is a 64×48 pixels gray scale image (Fig. 4). Therefore, the input layer has 64 by 48 neurons with values varying from 0.0, representing black, to 1.0, representing white.

The output layer consists of three neurons, one for each possible action on the environment: move left, move right, and shoot. Every output neuron represents the q-value for each action.

4) Summary: The network is illustrated in Fig. 3 and its topology is summarized in Table I.

TABLE I. NEURAL NETWORK TOPOLOGY.

Layer	Kernel	Stride	Filters	Input	Output
Conv1	4×4	2×2	32	64×48	32×24
Conv2	4×4	2×2	64	32×24	16×12
FC				12288	512
Output				512	3

D. Overview of a Single Step

In every execution step of the model, the views of the screens are pre-processed and passed to the networks of each agent. The network indicates which action should be taken. Both actions are then sent to the environment, which executes them and continues to the next step. This description is illustrated in Fig. 5.

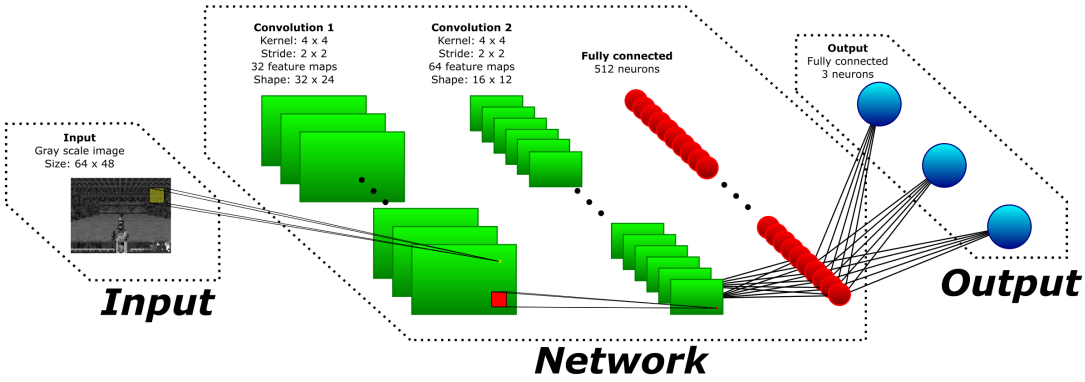


Figure 3. Summary of the neural network used in the controller of the agents.

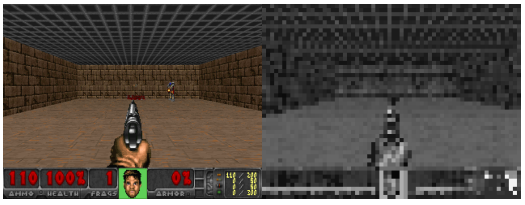


Figure 4. Screen (left) and pre-processed (right) images of agent's point of view.

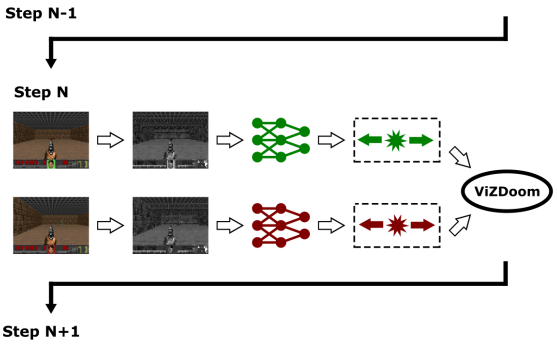


Figure 5. Layout of an execution step of the model.

V. Results and Discussion

A. Training

Fifteen agents were divided into three groups of five agents each. The groups were named Blue Team, Red Team, and Orange Team. The agents of Blue and Red Teams were trained in competition among them, in which each agent from Blue Team faced a single agent from Red Team. Thus, the agents of Blue and Red Teams learned their behaviors based on competition against other autonomous agents. The third group was trained by competing against opponents that act randomly, that is, they randomly take one of the three possible actions with equal probability (Fig. 6).

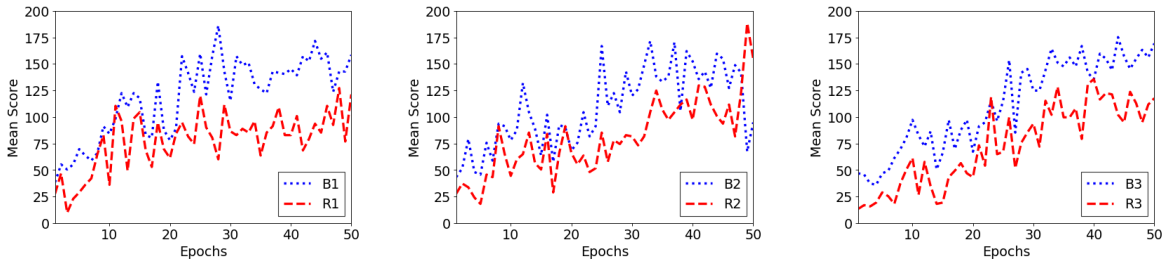
Every agent was trained during 50 epochs of 200 episodes each. An episode ends after 300 steps have passed. At the end of every episode, the agents are tested during 50 episodes, that is, they play the environment again, but only using the learned behaviors. All figures presenting the training results were taken from the test set. Before training, the agents are initialized with random weights according to the settings described in Section IV, which means that they do not have any previous knowledge about adequate behaviors, and making all of them different from each other.

1) Blue Team vs. Red Team: In this training, both agents are autonomous agents capable of learning, following the settings presented in Section IV. We can observe in all training results (Fig. 7) that all agents presented an ascending mean score curve. This indicates that all agents were capable of learning adequate behaviors to reach the goal of the proposed scenario: killing the opponent a great number of times. Thus, we can observe that training through competition between autonomous agents is adequate to learn behaviors inside a First-Person Shooter environment as presented in this work.

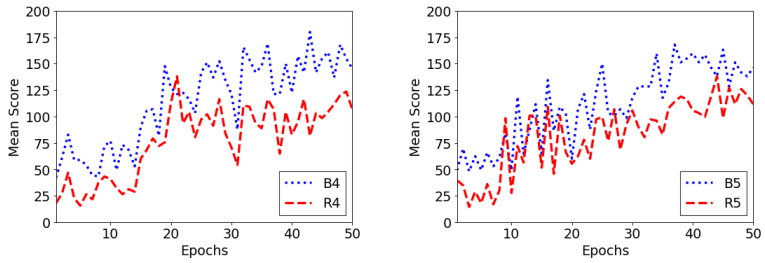
It is interesting to note that the agents from Blue Team had a greater performance over the agents of the Red Team during almost all training time. However, there was a single case in which an agent from Red Team was better at the end of training: agent R2 had a greater performance over agent B2. This happened because one of two possibilities: agent R2 had a fast and significant improvement or agent B2 showed a decrease in its behaviors. This case will be analyzed with more details in Section V-D.

The only difference in implementation of both teams is that agents of Blue Team are the servers of the multiplayer connection, and the agents of Red Team are the clients. To

Figure 6. Training scheme for each team.



(a) Agent B1 vs. Agent R1. (b) Agent B2 vs. Agent R2. (c) Agent B3 vs. Agent R3.



(d) Agent B4 vs. Agent R4. (e) Agent B5 vs. Agent R5.

Figure 7. Mean score of agents from Blue and Red teams during training. In every chart the dotted blue line indicates the mean score of an agent from Blue Team and the dashed red line indicates the mean score of an agent from Red Team.

evaluate this dynamic of connection, we alternate the server and the client after each epoch. This led to a result in which the server always had the best performance (Fig. 8). These results suggest that the servers may be receiving some kind of privileged information.

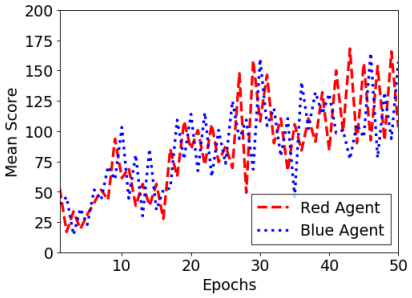


Figure 8. Results of alternating server and client players.

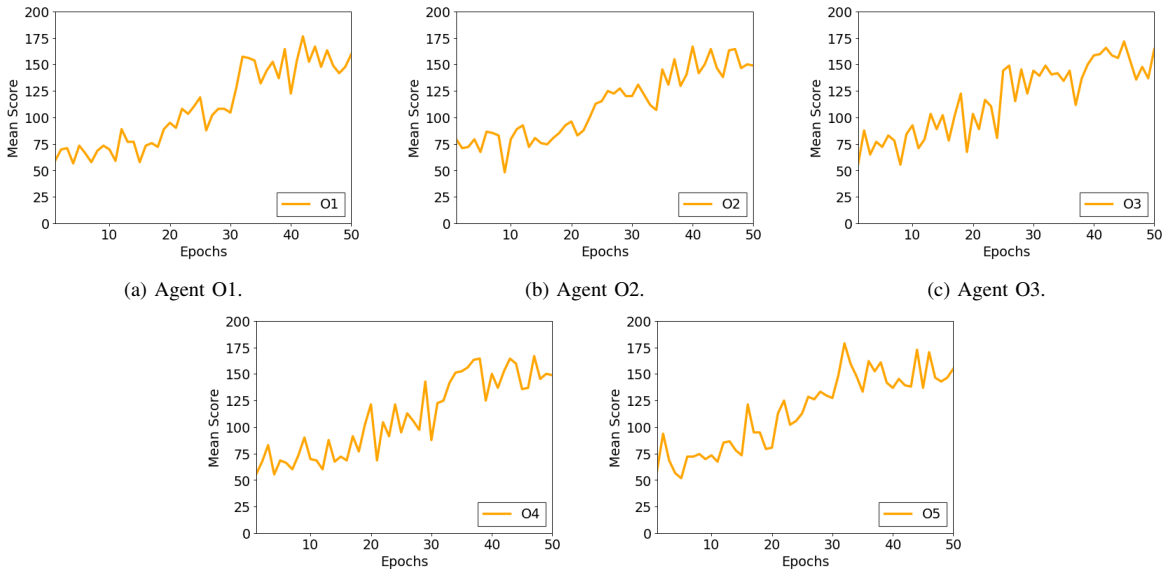
2) Orange Team vs. Random Opponents: The ascending curves observed in Fig. 9 indicates that learning occurred, that is, the agents were capable of learning adequate behaviors to play the environment. Moreover, in general, all agents finished the training with a mean score above 150, showing that they were capable to kill the opponents multiple times in a single episode. Thus, we can show that training by competing against opponents that act randomly is also adequate for an autonomous agent to learn in the presented scenario.

3) Consideration About Learning: We can observe by the ascending curves in training results (Fig. 7, Fig. 9), that all agents were capable of learning adequate behavior to compete in the test environment presented. Specifically, the agents learned behaviors like moving towards the opponent and shooting to kill (Fig. 10).

It is important to note that the same behavior of moving towards the direction of the opponent can lead to a kill (Fig. 10) or a death (Fig. 11). The only difference is how fast the shot is, whomever shoots first gains a kill score. This characteristic clearly shows that the difficulty of learning a behavior to increase the score can be harder than it looks.

In the case of Blue and Red teams, the agents of the two teams learned to move towards the opponent and shooting it when in front of it. When both agents move towards the opponent simultaneously, they are moving in opposing directions. Thus, if an agent shoots at the moment when they intersect, it will miss the shot, since the opponent will have already passed it when the bullet arrives. As we can see in Fig. 12, the agents were also capable of learning to anticipate the movement of the opponent, to kill it.

As both agents have a similar view of the same environment, it is expected that agents of all three teams develop similar behaviors. The behaviors described above were learned by agents from all teams. Specifically, the reaction speed and the accuracy of the shots were the main factors which defined the winner of an episode.



(a) Agent O1. (b) Agent O2. (c) Agent O3.

(d) Agent O4. (e) Agent O5.

Figure 9. Mean score of agents from Orange Team during training. In every chart the continuous orange line indicates the mean score of an agent from Orange Team.

.

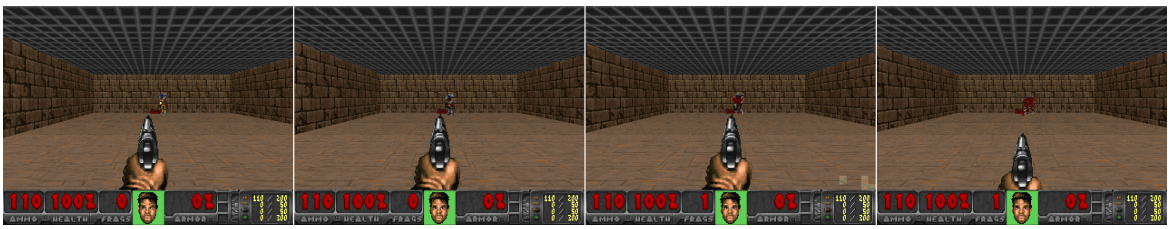


Figure 10. From left to right, **1st frame**: the agent starts in the left side of the room. **2nd frame**: it moves right, towards the opponent. **3rd frame**: it finds the opponent and shoots. **4th frame**: the opponent dies.

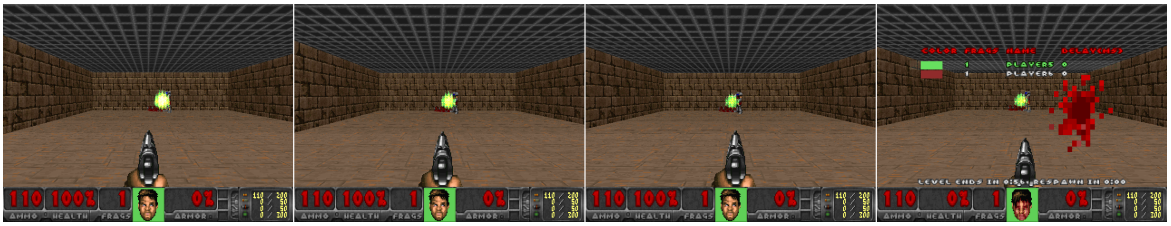


Figure 11. From left to right, **1st frame**: the agent starts in the left side of the room. **2nd frame**: it moves right, towards the opponent. **3rd frame**: it finds the opponent. **4th frame**: it dies before shooting the opponent.

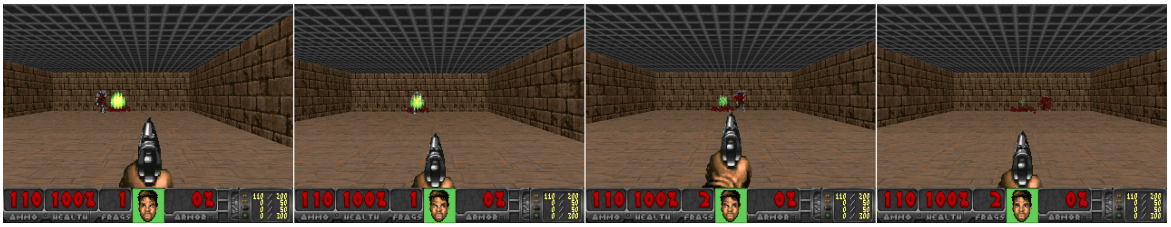


Figure 12. From left to right, **1st frame**: the agent starts in the right side of the room. **2nd frame**: it moves left, towards the opponent. **3rd frame**: it shoots in the limit of passing through the opponent. **4th frame**: the opponent dies.

B. Competition Against Random Opponents

Once all training is finished, the agents were tested in competition against opponents that act randomly. During the tests there was no learning. This is meant only for performance measurement, not for training.

Every agent of all teams played 200 matches against opponents that act randomly. In the end, the mean score of every agent was calculated. The Blue Team had a mean score of **113.64**, the mean score of Red Team was **78.60**, and the mean score of Orange Team was **156.84**.

We can see that Red Team had the worst performance, which is in accordance with the results showed in training scores (Fig. 7, Fig. 9). However, the agents from Orange Team had results much greater than Blue Team, as opposed to the performances shown along training.

Initially, this result could be considered unexpected, since we expected a performance improvement through competitive training between autonomous agents. However, there is a possibility that the Orange Team performed better because it was more used to random opponents. A detailed discussion will be made in Section V-D.

C. Competition Against Autonomous Agents

This time, a competitive environment between agents will be considered. Two autonomous agents from different teams are put in the scenario for a dispute between themselves. Again, there was no more learning at this moment, only the use of the network trained in Subsection V-A.

Every agent played 200 matches against all 10 agents from the other teams. In every match was established a victory result, if the agent had more kills than the opponent, a defeat result, if the agent killed less than the opponent, and draw, if both agents had the same number of kills. Afterwards, the percentage of victories was calculated for every matchup.

1) Blue Team vs. Red Team: After the training between these teams, Blue Team had better results, except for agent B2. In competition against random opponents, Blue Team also had better results. Thus, it is expected that Blue Team has a better result in tests against agents from Red Team.

In Table II we can see the percentages of victory for every agent in its match ups. The results are according to the expectations, since the performance of Blue Team was better than Red Team. The only exception was agent B2, which had a worse against all opponents. The case of agent B2 will be analyzed in Section V-D.

Thus, we show that autonomous agents which has a better performance against other autonomous agent during training will indeed have better results when competing with each other. Similarly, agents that have a worse performance after training, which is the case of agent B2, will also have worse results in tests.

2) Blue Team vs. Orange Team: When observing the results of training, we observed that Blue and Orange Teams had similar mean scores. Nevertheless, Orange Team had better results against random opponents. The next step is to verify if the performance of Orange Team was indeed better than the performance of Blue Team in a direct competition.

TABLE II. VICTORY PERCENTAGE IN EVERY COMPETITION BETWEEN THE AGENTS FROM BLUE AND RED TEAMS. THE LEFT PERCENTAGE REPRESENTS THE COLUMN AGENT (BLUE) AND THE RIGHT PERCENTAGE REPRESENTS THE ROW AGENT (RED). THE COMPLEMENT OF 100% SUM IS THE DRAW PERCENTAGE. IN BOLD ARE THE BEST PERFORMERS OF EVERY MATCHUP.

	B1	B2	B3	B4	B5
R1	**53,5%** × 31,0%	38,0% × **48,0%**	**60,5%** × 27,0%	**58,0%** × 30,5%	**57,0%** × 28,5%
R2	**53,5%** × 33,0%	33,0% × **56,5%**	**58,0%** × 29,0%	**57,5%** × 30,5%	**62,5%** × 29,0%
R3	**57,5%** × 25,0%	25,5% × **57,5%**	**54,5%** × 30,0%	**55,5%** × 27,0%	**56,0%** × 29,5%
R4	**53,0%** × 27,5%	30,0% × **57,0%**	**53,0%** × 31,0%	**53,5%** × 32,5%	**54,5%** × 33,0%
R5	**58,0%** × 27,0%	33,0% × **47,5%**	**50,0%** × 31,5%	**51,5%** × 32,0%	**56,0%** × 30,5%

The results in Table III show that agents from Blue Team had better results than the agents from Orange Team, except for agent B2. This agent will be analyzed in Section V-D.

Blue Team, even having an inferior performance against random agents, had better results than Orange Team in direct competition. Thus, we observe that training against other autonomous agents makes Blue Team perform better against other autonomous agents.

Since Blue Team had a superior performance over the Red Team during training, we can observe that agents which had better results in training against other autonomous agents will have a better performance against other intelligent agents. However, the training against random opponents does not guarantee a superior result against intelligent agents.

TABLE III. VICTORY PERCENTAGE IN EVERY COMPETITION BETWEEN THE AGENTS FROM BLUE AND ORANGE TEAMS. THE LEFT PERCENTAGE REPRESENTS THE COLUMN AGENT (BLUE) AND THE RIGHT PERCENTAGE REPRESENTS THE ROW AGENT (ORANGE). THE COMPLEMENT OF 100% SUM IS THE DRAW PERCENTAGE. IN BOLD ARE THE BEST PERFORMERS OF EVERY MATCHUP.

	B1	B2	B3	B4	B5
O1	**53,0%** × 37,5%	29,5% × **55,0%**	**49,5%** × 35,5%	**53,0%** × 33,5%	**48,0%** × 35,5%
O2	**46,0%** × 38,5%	31,5% × **54,5%**	**55,5%** × 31,0%	**60,0%** × 29,0%	**45,0%** × 39,0%
O3	**53,0%** × 29,0%	28,0% × **59,0%**	**48,5%** × 35,0%	**51,5%** × 35,0%	**48,5%** × 35,5%
O4	**51,5%** × 35,5%	28,0% × **59,5%**	**57,0%** × 31,5%	**54,5%** × 33,0%	**49,0%** × 34,5%
O5	**60,0%** × 26,0%	28,0% × **55,5%**	**52,5%** × 32,5%	**64,5%** × 25,5%	**57,5%** × 29,5%

3) Red Team vs. Orange Team: Both in training and in results against random opponents, the Red Team had an inferior performance in comparison to the Orange Team. Thus, it is expected that the Red Team also perform worse in a direct competition.

However, as can be seen in Table IV, the performance of the Red Team was better in all cases. Thus, we can observe that training against other autonomous agents, Blue Team, made Red Team perform better in a direct competition against autonomous agents trained with random opponents.

Since Red Team had an inferior performance than Blue Team during training, we can observe that agents which had

the worst results in competition against other autonomous agents will have a worse performance in competition against other intelligent agents. However, training against random opponents does not guarantee a superior performance against intelligent agents.

TABLE IV. VICTORY PERCENTAGE IN EVERY COMPETITION BETWEEN THE AGENTS FROM RED AND ORANGE TEAMS. THE LEFT PERCENTAGE REPRESENTS THE COLUMN AGENT (RED) AND THE RIGHT PERCENTAGE REPRESENTS THE ROW AGENT (ORANGE). THE COMPLEMENT OF 100% SUM IS THE DRAW PERCENTAGE. IN BOLD ARE THE BEST PERFORMERS OF EVERY MATCHUP.

	R1	R2	R3	R4	R5
O1	**52,0%** × 25,0%	**46,5%** × 34,0%	**56,0%** × 28,0%	**57,5%** × 29,5%	**43,5%** × 38,0%
O2	**58,5%** × 23,0%	**56,0%** × 29,0%	**50,0%** × 38,0%	**51,5%** × 34,0%	**43,5%** × 34,0%
O3	**55,5%** × 28,0%	**55,0%** × 26,5%	**55,0%** × 29,0%	**50,0%** × 33,5%	**43,5%** × 38,5%
O4	**47,0%** × 28,5%	**40,0%** × 37,0%	**48,0%** × 27,0%	**58,5%** × 29,5%	**50,5%** × 33,5%
O5	**56,0%** × 27,0%	**61,5%** × 25,0%	**58,0%** × 29,0%	**56,5%** × 27,0%	**49,5%** × 39,5%

D. Final Discussions

1) Agent B2: During training between Blue and Red teams, the only case in which an agent from Blue Team finished the training with a mean score less than a Red agent was agent B2. In Fig. 7b, we can see a sudden drop in the results of the last epochs. Two possibilities will be evaluated: agent B2 had a significant decrease in its behaviors or agent R2 had an increase in relation to its opponent.

In the tests made in Subsection V-C, the agent B2 was also the only exception in comparison with other agents from Blue Team. Since B2 had worse results in all tests and agent R2 did not show a significant improvement in the tests in relation to other agents from Red Team, we observe that a degeneration occurred on the behavior of agent B2.

2) Better Performance of Orange Team Against Random Opponents: When trained in an environment against opponents that act randomly, the Orange Team had the best results. Since Blue and Orange Teams had similar mean scores, we expected that both had similar results also in this test. To explain this difference, two possibilities arise: the agents from Orange Team learned indeed better behaviors, or they just learned how to respond to random opponents?

The results of the tests against Blue Team (Table III) show that the agents from Orange Team had worse results, except the agent B2. Thus, the second hypothesis gains strength, since we cannot say that the agents from Orange Team learned better behaviors than the agents from Blue Team.

Analyzing the results against Red Team, we have even stronger evidence. Comparing the mean score results during training and the mean score against random opponents, the Orange Team clearly had an advantage over Red Team. However, in direct competition, the Red Team had better results in all cases. These facts further strengthen that the performance of agents from Orange Team was better because they were trained against random opponents.

3) Competition and Intelligence: The goal of this work is to evaluate if training through competition against autonomous agents leads to better results than the training against non-intelligent agents. Since the agents from Blue Team had better results in direct competition against agents from Orange Team, we can state that agents which train against intelligent agents **and** can present better results, having a better performance in direct competition than agents trained against non-intelligent agents.

Since Red Team, even having a general performance worse than Blue and Orange teams during training and against random opponents, had better results than Orange Team in direct competition, we can say that agents which train against intelligent opponents, will have better results in direct competition than agents that trained against non-intelligent opponents.

VI. CONCLUSION

This paper presents an experimental evaluation of training between autonomous agents against two different types of opponents, other autonomous agents, and non-intelligent agents. To achieve this goal, we used a Deep Q-Network model, which was trained to learn how to play a competitive custom scenario of ViZDoom. The training results showed that all agents learned adequate behaviors and kill the opponent multiple times in a single episode. They also learned behaviors like moving towards the opponent and shooting at it, and very complex behaviors like anticipating the opponent's movement. Thus, these agents are able to be tested in competition against other agents.

The main goal of this work was to show that training through competition against autonomous agents improves the performance of agents over training against non-intelligent agents. The three teams of agents were trained against two different types of opponents. Two teams trained against autonomous agents and one team trained against random opponents. The tests executed for agents who trained through competition against autonomous agents clearly showed a better performance over agents trained by competing against non-intelligent agents. The results indicate that, although training against non-intelligent agents has advantages, training by competing against other autonomous agents will lead to a better performance.

A. Future Works

This work serves as a guideline for evaluating competitive training between autonomous agents. Some features could be modified to evaluate new aspects of this type of training, such as training one agent against all the agents of another team, allowing competition between agents of the same team, creating different types of non-intelligent agents, and modifying specific aspects of the environment, like adding hitpoints for every agent. All of these points can lead to

interesting behaviors and have an effect on the performance of the agents. Some other suggestions are presented below.

1) Testing Different Architecture Settings: The controller presented in this work does not limit the architecture to the settings used. Choosing different architecture settings is not only possible, but can lead to interesting results. However, these changes will affect the basis of the agents involved and all training needs to be done from the beginning, which will take time. Particularly, increasing the size of the neural network could be beneficial, since more features could be learned and passed throughout the layers. However, a larger network leads to greater complexity, which will decrease learning speed, as well as training will be harder and could make learning unfeasible.

2) Adding External Information: One characteristic of the agents presented in this work is the ability to learn receiving as input only the pixels of the screen. There are models that give additional information to the agents [3], [20] and can improve their behaviors. However, augmented information should be used with care, because it is interesting that an agent learns some features by itself, without the need for external manipulation.

3) Using Different Types of Agents: ViZDoom's environment needs as input only the actions corresponding to keyboard and mouse activities. Thus, agents that do not use neural networks could also be evaluated in the game. The needed adaptations are specific for each case and the results will not necessarily be the same. Additionally, comparing agents with different types of controllers can raise interesting questions to be analyzed.

4) Verifying the Problems of Agent B2: As seen in Subsection V-D1, a single agent presented very different results than the others. The review of the results and the source code indicates that this problem can be related to internal dynamics of the environment, since all agents have almost the same implementation. To solve this problem, a more specific analysis should be performed.

ACKNOWLEDGMENT

The authors would like to thank CNPq, the National Council for Scientific and Technological Development, and the Graduate program (MS and PhD) in Computer Science at the Federal University of Ceará (MDCC/UFC) for their financial support.

REFERENCES

[1] S. Alvernaz and J. Togelius, "Autoencoder-augmented neuroevolution for visual doom playing," *CoRR*, vol. abs/1707.03902, 2017. [Online]. Available: http://arxiv.org/abs/1707.03902

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[3] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. S. Torr, "Playing doom with slam-augmented deep reinforcement learning," *CoRR*, vol. abs/1612.00380, 2016. [Online]. Available: http://arxiv.org/abs/1612.00380

[4] A. O. Castañeda, "Deep reinforcement learning variants of multi-agent learning algorithms," Master's thesis, School of Informatics University of Edinburgh, 2016.

[5] Z. Chen and D. Yi, "The Game Imitation: A Portable Deep Learning Model for Modern Gaming AI," CS231n: Convolutional Neural Networks for Visual Recognition, 2016. [Online]. Available: http://cs231n.stanford.edu/reports/2016/pdfs/113_Report.pdf

[6] A. Das, S. Kottur, J. M. F. Moura, S. Lee, and D. Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2970–2979.

[7] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *CoRR*, vol. abs/1611.01779, 2016. [Online]. Available: http://arxiv.org/abs/1611.01779

[8] M. Egorov, "Multi-Agent Deep Reinforcement Learning," CS231n: Convolutional Neural Networks for Visual Recognition, 2016. [Online]. Available: http://cs231n.stanford.edu/reports/2016/pdfs/122_Report.pdf

[9] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 2137–2145.

[10] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate to solve riddles with deep distributed recurrent q-networks," *CoRR*, vol. abs/1602.02672, 2016. [Online]. Available: http://arxiv.org/abs/1602.02672

[11] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323.

[13] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems*, G. Sukthankar and J. A. Rodriguez-Aguilar, Eds. Cham: Springer International Publishing, 2017, pp. 66–83.

[14] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, pp. 947 EP –, Jun 2000. [Online]. Available: http://dx.doi.org/10.1038/35016072

[15] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," in *Proceedings of the International Conference on Learning Representations (ICLR)*, May 2016, pp. 1–12.

[16] M. J. Hausknecht, "Cooperation and communication in multi-agent deep reinforcement learning," Ph.D. dissertation, Graduate School of The University of Texas at Austin, 2016.

[17] H. He, J. Boyd-Graber, K. Kwok, and H. D. III, "Opponent modeling in deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1804–1813.

[18] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, "Vizdoom: A doom-based AI research platform for visual reinforcement learning," *CoRR*, vol. abs/1605.02097, 2016. [Online]. Available: http://arxiv.org/abs/1605.02097

[19] X. Kong, B. Xin, F. Liu, and Y. Wang, "Revisiting the master-slave architecture in multi-agent deep reinforcement learning," *CoRR*, 12 2017.

[20] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," *CoRR*, vol. abs/1609.05521, 2016. [Online]. Available: http://arxiv.org/abs/1609.05521

[21] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 464–473. [Online]. Available: http://dl.acm.org/citation.cfm?id=3091125.3091194

[22] Y. Liang, M. C. Machado, E. Talvitie, and M. Bowling, "State of the Art Control of Atari Games Using Shallow Reinforcement Learning," *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 485–493, 2016.

[23] L.-J. Lin, "Reinforcement learning for robots using neural networks," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 1993, uMI Order No. GAX93-22750.

[24] M. McKenzie, P. Loxley, W. Billingsley, and S. Wong, "Competitive reinforcement learning in atari games," in *AI 2017: Advances in Artificial Intelligence - 30th Australasian Joint Conference, Melbourne, VIC, Australia, August 19-20, 2017, Proceedings*, 2017, pp. 14–26. [Online]. Available: https://doi.org/10.1007/978-3-319-63004-5_2

[25] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *arXiv*, vol. 48, pp. 1–28, 2016. [Online]. Available: http://arxiv.org/abs/1602.01783

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. a. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[28] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," *CoRR*, vol. abs/1707.04402, 2017. [Online]. Available: http://arxiv.org/abs/1707.04402

[29] E. Parisotto, L. J. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *CoRR*, vol. abs/1511.06342, 2015. [Online]. Available: http://arxiv.org/abs/1511.06342

[30] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," *CoRR*, vol. abs/1705.05363, 2017. [Online]. Available: http://arxiv.org/abs/1705.05363

[31] J. Romoff, E. Bengio, and J. Pineau, "Deep Conditional Multi-Task Learning in Atari," *ICML 2016*, vol. 48, 2016.

[32] P. B. S. Serafim, Y. L. B. Nogueira, C. A. Vidal, and J. B. Cavalcante-Neto, "On the development of an autonomous agent for a 3d first-person shooter game using deep reinforcement learning," in *Anais do XVI Simpósio Brasileiro de Jogos e Entretenimento Digital*, 2017.

[33] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. [Online]. Available: http://dx.doi.org/10.1038/nature16961

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[35] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[36] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *CoRR*, vol. 12, 11 2015.

[37] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," Coursera: Neural Networks for Machine Learning, 2012.

[38] H. van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," *ArXiv e-prints*, Jun. 2017.

[39] Y. Wu and Y. Tian, "Training agent for first-person shooter game with actor-critic curriculum learning," in *International Conference on Learning Representations (ICLR)*, 2017, pp. 1–10.