Journal of Intelligent & Robotic Systems (2021) 102: 69
https://doi.org/10.1007/s10846-021-01333-1

Deep Reinforcement Learning for a Humanoid Robot Soccer Player

Isaac Jesus da Silva[1] · **Danilo Hernani Perico**[1] · **Thiago Pedro Donadon Homem**[2] · **Reinaldo Augusto da Costa Bianchi**[1]

Received: 29 September 2020 / Accepted: 27 January 2021 / Published online: 26 June 2021

Abstract

This paper investigates the use of Deep Reinforcement Learning (DRL) applied to the humanoid robot soccer environment, where a robot must learn from basic to complex skills while it interacts with the environment through images received by its own camera. To do so, the Dueling Double DQN algorithm is used: it receives the images from the robot's camera and decides on which discrete action should be performed, such as walk forward, turn to the left or kick the ball. The first experiments were performed in a robotic simulator in which the robot could learn, with DRL, three different tasks: to walk towards the ball, to act like a penalty taker and to act like a goalkeeper. In the second experiment, the learning obtained in the task to walk towards the ball was transferred to a real humanoid robot and a similar behavior could be observed, even though the environment was not exactly the same when the domain was changed. Results showed that it is possible to use DRL to learn tasks related to the role of a humanoid robot-soccer player, such as goalkeeper and penalty taker.

Keywords Deep reinforcement learning · Humanoid robots · Robot cognition · RoboCup soccer competition

1 Introduction

Many researches [1–4] have shown important improvements in the development of robots that can act autonomously. Generally, all these robots use software architectures composed of several processes, where each process is responsible for a specific task, such as object detection, localization, path planning, decision making and so on. However, the problem is that all those robot software must be developed and configured by the researcher, and, in some cases, it does not reach the optimal solution for the task to be solved. Moreover, these complex architectures need a several quantity of manual setup and synchronization between processes.

A possible solution to replace or reduce the complexity of all these kind of software architectures is the use of Reinforcement Learning or, more precisely, its enhanced version - Deep Reinforcement Learning (explained in Section 2) - thus the robot itself can learn the task assigned to it by receiving rewards while it interacts with the environment.

Deep Reinforcement Learning algorithms has been successfully applied in 3D games [5–10] allowing virtual agents to learn how to take decisions in three-dimensional environments from game images. Besides games, DRL has been also used for the development of some cognitive tasks in robotic agents [11–13], allowing the robots to learn tasks such as object manipulation.

Thus, inspired by the learning applied for games and cognitive robots, this paper aims to investigate the usage of Deep Reinforcement Learning as a way of providing to the robots the ability to learn tasks related to the humanoid robot soccer domain just by observing the environment through images from their own camera. As there are many possible soccer player roles and positions on the field, two distinct roles were chosen to be learned by the robot: goalkeeper and penalty taker. Therefore, the objective of this work is to empower a humanoid robot soccer with learning abilities, by means of Deep Reinforcement Learning, thereby the robot itself must be able to learn the best actions for performing

✉ Isaac Jesus da Silva
isaacjesus@fei.edu.br

Danilo Hernani Perico
dperico@fei.edu.br

Thiago Pedro Donadon Homem
thiagohomem@ifsp.edu.br

Reinaldo Augusto da Costa Bianchi
rbianchi@fei.edu.br

[1] FEI University Center, 3972-B Humberto de Alencar Castelo Branco Ave., Assunção, São Bernardo do Campo, SP, 09850-901, Brazil

[2] Federal Institute of Education, Science and Technology of São Paulo, 951 Mutinga Ave., Jardim Santo Elias, São Paulo, SP, 05110-000, Brazil

a task, without the need of being manually programmed for each process.

However, applying Deep Reinforcement Learning in real robots is still a very hard task, since this kind of learning needs a huge amount of steps, which means a lot of interaction with the environment and, consequently, lots of time, until some learning can be achieved.

An alternative is to perform the learning in a simulator and reuse the learned model on a real robot. One main advantage in the usage of a robotics simulator is the possibility to perform the learning without damaging the robots. Besides that, most simulators allow the acceleration of the simulation, which saves time.

This paper is structured as follows: Section 2 provides the basic concepts of Deep Reinforcement Learning. Section 3 depicts related work. Section 4 presents the proposal of this research. Section 5 describes the experiments and results in which: Section 5.2 the robot learns how to act as a goalkeeper; Section 5.3 the robot learns how to act as a penalty taker; and Section 5.4 the network weights learned by the robot on simulator is transferred to a real robot. Section 6 provides the conclusions.

2 Deep Reinforcement Learning

In reinforcement learning, the agent learns through interaction with the environment, receiving negative or positive rewards, according to the actions taken. In this model, the agent is expected to learn a policy that maximizes the reward accumulated over time [14]. The reinforcement learning problems are formalized as Markov Decision Processes. A MDP is composed of a set of states $s \in \mathcal{S}$, a set of actions $a \in \mathcal{A}$ to each state s, a transition model $P(s'|s,a)$ and a reward function $R(s,a,s')$ [15], where: $\mathcal{S}$ is a set finite of state; $\mathcal{A}$ is a set finite of action and $a(s)$ are the actions allowed for the state s; $P(s'|s,a)$ or $T(s,a,s')$ is the transition function; $R(s,a,s')$ is the reward get s' by performing the action a being the state s other reward $R(s')$ only the resulting state.

In deep reinforcement learning (in some models) the states are represented by features extracted from the images observed by the agent, for that, convolutional neural networks are used to extract such features. Deep Reinforcement Learning can learn policies directly from high-dimensional inputs, being an approach that applies RL from end to end [16].

In 2010 Lange, Sascha and Riedmiller [17] presented a deep reinforcement learning algorithm using deep autoencoders that learned action policies by mean purely visual input. They tested the algorithm in a grid-world of 30x30 pixels, the agent must learn to reach the goal from any position in the world, and the agent can visit any position in the space of 30x30 pixels (the agent size is 5x5 pixels). This model uses the deep autoencoders to learn a low dimensional representation of the input image, and the reinforcement learning algorithm (NFQ [18]) learns from state space provided by deep autoencoders. The difference between DQN and the deep autoencoders with NFQ, is that DQN updates the network weights by learning the relevant features in relation to the actions taken and the reward received, being an approach that applies RL from end to end [16].

In 2013 researchers from DeepMind (now known as Google DeepMind) released an article called *Playing Atari with Deep Reinforcement Learning* in Arxiv[1] [19]. They presented a Deep Reinforcement Learning model that learns the actions policies of a game using as input the game images. They demonstrated that the computer was able to learn to play some Atari games just from the raw frames of the game, and receiving the reward by the game's score. In this article were tested seven Atari games.

In 2015, this same group (Google DeepMind researchers) presented an article [16] showing that in Atari games the Deep Reinforcement Learning model called Deep Q-Network (DQN) outperformed all previous algorithms, being able to outperform the level of a professional human player in some games. The DQN differs from deep autoencoders with NFQ for being an approach that applies reinforcement learning end-to-end, directly from the visual inputs [19]. DQN updates the network weights by learning the relevant features in relation to the actions performed and the reward received.

Volodymyr Mnih et al. [16, 19] have proposed new algorithms based in the DQN aiming at the improvement of the reduction of the training time [8, 20] and in the game score performance [21, 22]. New applications for deep reinforcement learning also arise such as in continuous control [23], and also being applied to different games [5–9]. For instance, Jaderberg et al. [7] use deep reinforcement learning in a 3D maze game.

The DQN uses as input only the frames of the game, and the output are the possible actions for the game. The network is composed for some convolutional neural network layers followed by two fully connected network layers, and there is no layer of pooling.

$$y_j^{DQN} = r_j + \gamma \max_a Q(s_{j+1}, a; \theta_i^-) \quad (1)$$

where: s_{j+1} is the preprocessed last 4 frames (time steps $j+1$) of the video to represent the state; a is a set finite of action; θ are the weights of the network; $\theta_i^- = \theta_{i-1}$, parameters from the previous iteration θ_{i-1} are use to optimising the loss function $L(\theta_i)$; r is the reward; γ is the discount factor; For the update of weights θ, y_j^{DQN} is

[1] https://arxiv.org

the target for the evaluation the of network output. During the interaction of the agent with the environment, in order to maximize the accumulated reward over time, the agent must select optimal value functions, satisfying the Bellman Equation according to the Eq. 1.

$$L(\theta_i) = \frac{1}{n}\sum_{j}^{n}\left[(y_j^{DQN} - Q(s_j, a_j; \theta_i))^2\right] \quad (2)$$

To train the network by adjusting the weights values θ by the back-propagation, is performed a gradient descent (3) called RMSProp applied to L function for each iteration i according to Eq. 2, where $\theta_i^- = \theta_{i-1}$ and γ is the discount factor. RMSProp adjusts step-size parameter for each weight running average of its recent magnitude to accelerates learning [24].

$$\nabla_{\theta_i} L(\theta_i) = (y^{DQN} - Q(s, a; \theta_i))\nabla_{\theta_i} Q(s, a; \theta_i) \quad (3)$$

The agent uses a technique known as experience replay to store experiences [25], therefore during the game all experiments $e_t =< s_t, a_t, r_t, s_{t+1} >$ are stored in a variable called replay memory $\mathcal{D} = \{e_1, e_2, \ldots, e_n\}$.

2.1 Double DQN

Hasselt [26] presented an algorithm called Double Q-learning to solve the problem of the Q-learning algorithm that overestimates the Q value function under certain conditions. So, researchers at Google [21] found that in certain Atari games, this effect of overestimating the Q value function occurs more sharply. Then, these researchers created the Double DQN algorithm based on the algorithm Double Q-learning [26]. With Double DQN in Atari games, it was possible to increase the score in games compared to DQN using the same parameter values, so Double DQN presents better policies than DQN for Atari games [21].

2.2 Prioritized Experience Replay

In DQN, the use of experience replay stored in replay memory $\mathcal{D}$ allows agents to remember and reuse experiences from the past. Experience replay can reduce the amount of experience the agent needs to learn. However, the use of these samples stored in replay memory $\mathcal{D}$ is being handled in the same way, not being considered that the agent can learn more of some transitions than others.

In the Schaul [20] article, researchers have developed a method that can make learning from experience replay more efficient, have developed a way of prioritizing experience, thus reproducing important and efficient transitions more often. The main goal of prioritized replay is the importance of measuring each transition, assigning a priority to the transition.

2.3 Dueling Double DQN

Wang, de Freitas and Lanctot [22] developed a deep reinforcement learning architecture named Dueling Double DQN (also called Dueling Network Architecture). This architecture has two parallel networks of fully connected neurons representing the value function $V(s)$ and the advantage function $A(s, a)$, since the output combines these two functions producing the state-action value function $Q(s, a)$ ($Q(s, a) = V(s) + A(s, a)$), but they share the same layers of convolutional networks. The advantage function specifies how good it is for the agent to perform an action compared to other actions. The Dueling Network Architecture is an enhancement of the Double DQN and Prioritized Experience Replay algorithms without imposing any changes to these algorithms.

3 Related Work

In the article of Tai and Liu [27], the authors used DQN so that a mobile robot can learn to navigate an environment by avoiding obstacles, these same authors previously developed some work with Deep Learning in mobile robot navigation [27–29]. The authors believed that it was the first time that a mobile robot developed navigational capability through deep end-to-end reinforcement learning using only raw sensor information (without any pre-processing) [27]. In this case the sensor used was a *kinect* which supplies camera data in RGB-D.

In the article of Lobos-Tsunekawa et al. [30], the authors propose a map-less visual navigation system for biped humanoid robots in robotic soccer. The authors use the Deep Deterministic Policy Gradients (DDPG) algorithm, which corresponds to an actor-critic DRL algorithm. The network is composed by convolutional layers, fully connected layers and Long Short-Term Memory (LSTM) layers.

In the article of Abreu et al. [31] the authors use the Proximal Policy Optimization (PPO) algorithm to optimize the skills of running and dribbling the ball, have the aim to achieving natural gaits without sacrificing performance in a humanoid robot soccer. They performed the experiment in Simspark robot simulator to simulate a NAO humanoid robot playing soccer. In this work they do not use the robot's camera, acting directly on the 20 joints of the robot.

Some works [11–13] demonstrated that a real manipulative robot with deep reinforcement learning can learn some object manipulation tasks or even open a door. In the article of Gu et al. [13] demonstrated that with deep reinforcement learning the robot can learn to perform tasks. The article shows the manipulator robot learning to perform a complex task, such as opening a door, without having to present manual demonstrations or representations.

Some works with Deep Reinforcement Learning showed the DRL ability to learn to play games in 3D environments [5–9]. These works showed the DRL learning to make decisions in three-dimensional environments from the game images.

In the work of Michał Kempka et al. [5], the researchers developed a version of the video game Doom called VizDoom to be used as a platform for testing machine learning algorithms in a game with a three-dimensional environment. Doom is a first-person shooter. The researchers performed two experiments with DRL: the first experiment, the agent has to learn to shoot in the personages that appear; in the second experiment, the agent has to learn to navigate in a more complex maze.

In the work of Piotr Mirowski [9], the researchers tested DRL in a 3D maze game, showing that the agent is able to learn to navigate in the environment, even in conditions where the location of the objective changes frequently. In this work, the authors provide a detailed analysis of the agent's behavior and its ability to localizate itself, showing that the agent learns the main navigation skills in this game.

4 Deep Reinforcement Learning for a Humanoid Robot Soccer Player

In order to perform any task, most robots use systems composed of several processes, such as: vision recognition to detect objects; control process to perform actions; localization system by which the robot itself can discover where it is located; and decision process, responsible for choosing the robot's actions given the sensory system. However, all these processes need manual and, sometimes, empirical setup of their parameters to work as expected. Furthermore, these processes must be synchronized, which can be a very stressful and hard job. Figure 1 shows two examples of software architecture for humanoid robots inside the soccer domain. As it can be seen, CITBrains [32] and RoboFEI [4] have several processes.

Thus, this paper proposes the usage of the Deep Reinforcement Learning approach to replace almost all the processes the robot usually needs by only two: the DRL process that will determine actions for the robot, encompassing skills related to the vision system, localization and decision; and the control process to perform the chosen actions. Figure 2 depicts the proposal, where the tasks are learned by the interactions with the environment and only two processes are needed: DRL process substitutes vision, localization and decision processes.

The DRL shown in Fig. 2 has as input only the frames x from the robot camera (environment observation). The network output are the possible actions a to be performed, where the types of actions are configured according to

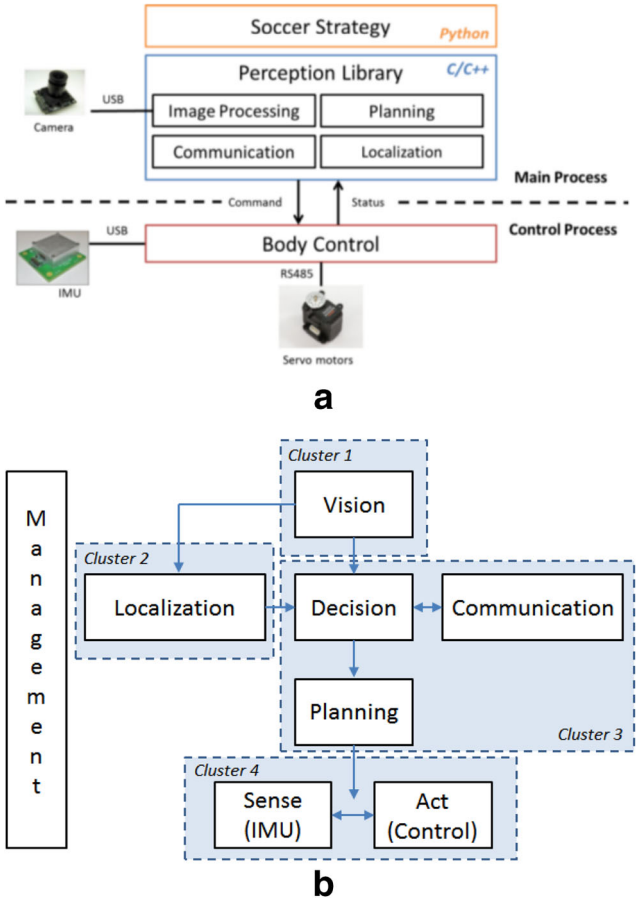

Fig. 1 Examples of software architecture for humanoid robots. **a** CITBrains [32]. **b** RoboFEI [4]

the role to be learned. The network implemented in this model consists of some convolutional neural networks layers followed by two fully-connected networks layers.

The convolutional neural networks layers are responsible for learning the features that represent the states s of the model. Therefore, the state s_t is considered as a sequence of actions and observations (where in this domain each action and observation is a step) $s_t = x_1, a_1, x_2, a_2, x_3, a_3, \ldots, x_t$, considering t as a finite time of these sequences (a sequence of actions and observations is used so that the DRL learns the dynamics of the environment). Thus, the model have a large and finite Markov Decision Process (MDP) [33] in which each sequence is a distinct state s_t.

The technique known as *experience replay* [25] is used to store the agent's experiences, so during the learning the experiences $e_t =< s_t, a_t, r_t, s_{t+1} >$ are stored in a variable called *replay memory* $\mathcal{D} = \{e_1, e_2, \ldots, e_n\}$. During the learning are used small amounts of the experiences e_t to prevent the network from moving to a local minimum, these experiences are random selected from the experiences stored in *replay memory* $\mathcal{D}$.

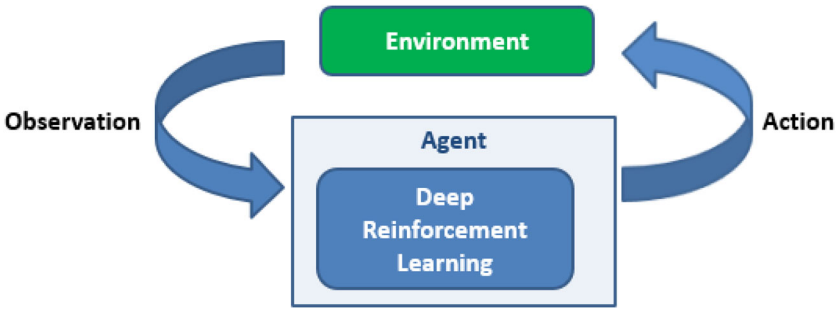

Fig. 2 Proposal of a new approach where a DRL process substitutes vision, localization and decision processes

Table 1 Model architecture Dueling Double DQN

Layer	Input	Kernel	Stride	N. Kernels	Output
Conv1	84x84xξ	8x8	4	32	20x20x32
Conv2	20x20x32	4x4	2	64	9x9x64
Conv3	9x9x64	3x3	1	64	7x7x64
fc1	7x7x64			512	512
fc2	512			*a*	*a*

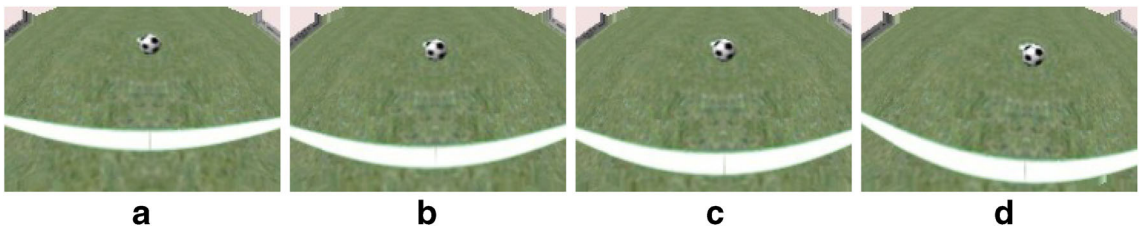

Fig. 3 Image from the simulated robot camera

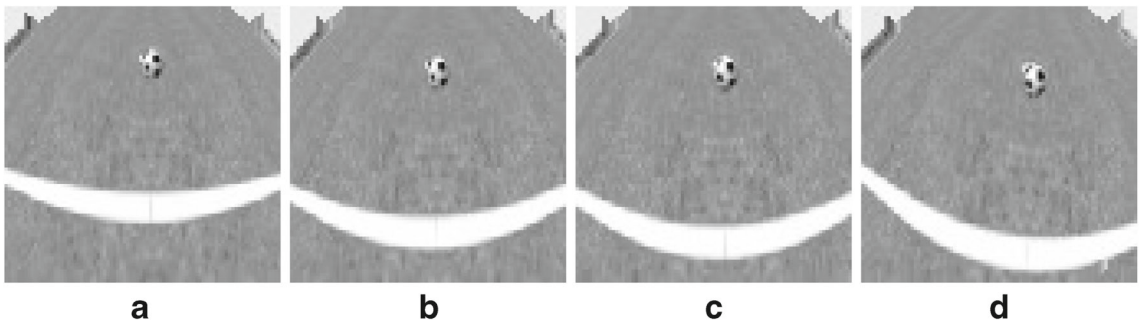

Fig. 4 Input images 84x84

Fig. 5 Goalkeeper robot

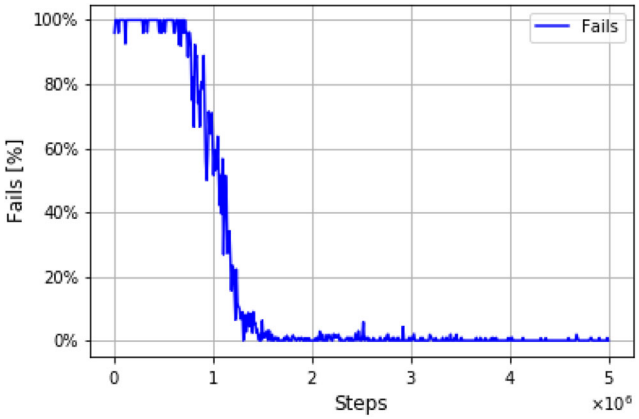

Fig. 7 Percent of fails per 10000 steps

In the learning process the DRL perform episodes until it learns the proposed role (an episode consists of a sequence of steps from the beginning to the end of the task). The agent's aim is learns through interaction with the environment, receiving negative or positive rewards, according to the actions taken, maximizes the reward accumulated over time. The reward was modeled according to the role that the robot must perform, where the rewards will be presented in the Section 5.

More precisely, the focus of this research is to investigate the usage of DRL in a humanoid robot soccer player, as a way of providing to the robot the ability to learn tasks just by observing the environment through images from its own camera. Thereby, two different roles were chosen to be learned by the robot in a robotic simulator: "to act like a goalkeeper" and "to act like a penalty taker". Besides, a case study was performed where the robot is expected to learn how to "walk towards the ball" in the simulator and, then, the learned network weights are transferred to a real robot.

Inspired by the learning in 3D games (Section 3), the DRL algorithm used in this research is the Dueling Double DQN [22], where the algorithm is responsible for receiving the images from the robot's camera and for performing one discrete action, such as walk forward, turn to the left, kick the ball and so on. The set of actions vary depending on the role that should be learned. As the robot execute the actions and interacts with the environment, it receives rewards. When the robot learns the whole policy of actions, it learns how to perform a full given task.

One main difference between the domain of the 3D games and the humanoid robot soccer environment is the fact that games normally have deterministic actions, besides having little variation in lighting and colors. On the other hand, the humanoid robot soccer game is more stochastic, even when executed in a simulator.

The Dueling Double DQN architecture used in this work can be seen in Table 1. The input consists of ξ matrices (images) of 84x84. It uses ξ sequential frames in order to allow the capture of the environment's dynamics. The temporal difference between these images is τ; therefore, the network receives an image referring to the current instant and $\xi - 1$ past images. The original camera image (RGB format) is transformed into the YUV format, using only the Y (luminance) channel (Images from Y channel

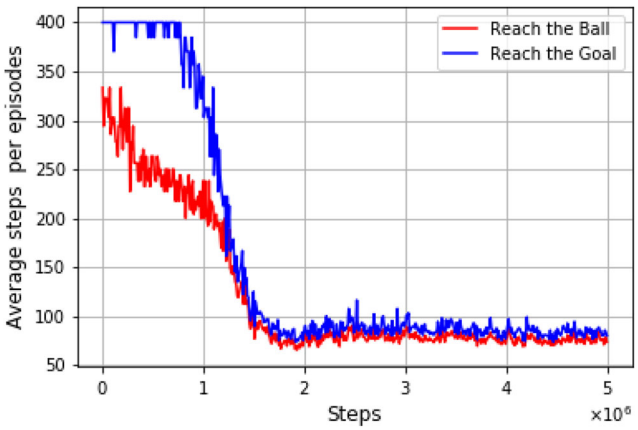

Fig. 6 Average steps per episode

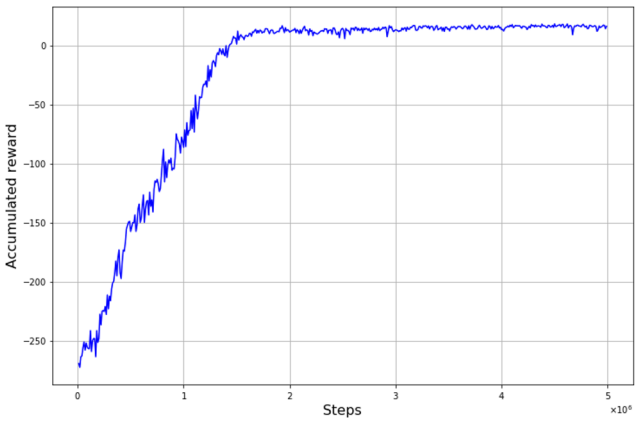

Fig. 8 Accumulated Reward

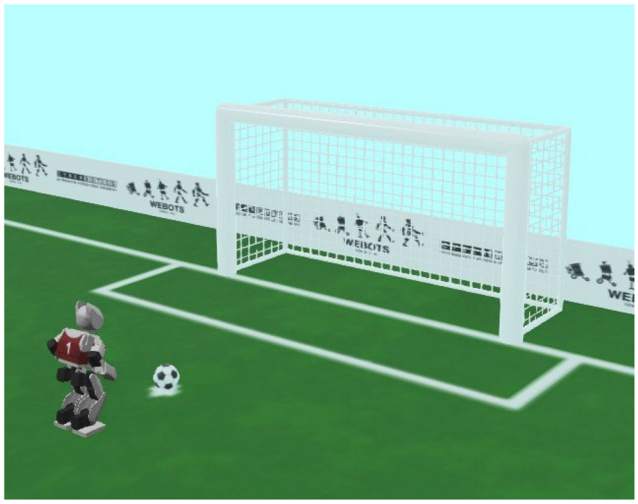

Fig. 9 Scenario of running the penalty taker experiment

presents images in gray-scale), and resizing the frame to 84x84.

In this research two models of humanoid robots were used: the real robot developed at the Centro Universitário FEI and the simulated humanoid robot called DARwIn-OP [34, 35] available on the Webots robot simulator.

5 Experiments and Results

This section presents three different experiments performed with our DRL proposal model: 1) on simulator, the robot-soccer learns to perform a goalkeeper role; 2) on simulator, the robot-soccer learns to perform a penalty kick role and 3) on simulator and with a real-robot, the robot-soccer learns to walk towards the ball in the simulator and the network weights learned on simulation were transferred to the real robot-soccer.

5.1 Initial Considerations

Tests were performed on simulated and real robot-soccer environments aiming to show that a DRL model is able to substitute processes like Vision, Decision and Localization (Fig. 1b). With these experiments we want to show that it is possible for a DRL model to replace all those processes (Fig. 2). Therefore, a single DRL process can replace three processes: vision (responsable for objects detect); decision (responsible for collects informations from the localization, vision and communication processes and takes a decision about what to do); localization (localization is in charge of the self-localization of the robot in the field), and each of these processes uses specific algorithms that are presented in the article by Perico et al. [4].

Simulated experiments were conducted in the Webots simulator[2] [36]. Webots is a robotic simulator that offers a three-dimensional virtual environment with physical properties. The robot used in the simulator was the DARwIn-OP humanoid robot [34, 35]. The environment was setup like a soccer field. Both robot and environment are available in the Webots simulator. The simulated experiments used a white ball, white goal and the field containing the dimensions of 9x6 meters according to the rules[3] of 2018 RoboCup humanoid league.

The real robot-soccer used in the experiments was developed by Centro Universitário FEI, which have their mechanics based on DARwIn-OP [37]. The robot has 49 cm of height, 3 Kg of weight, and 20 servo-motors Dynamixel RX-28. As sensors, it uses an UM7 Ultra-Miniature Orientation Sensor and a Logitech HD Pro Webcam C920 (Full HD). And the computer used is an Intel NUC[4] Core i5-4250U, 8GB SDRAM, 120GB SDD [4, 38]. In order to increase the field of view of the camera and eliminate the pan and tilt servo motors, the robot camera is equipped with a fish-eye lens.

All the images are directly taken from the robot's camera. Then, images are reduced to 84x84 and converted to gray scale color. All experiments use $\xi = 4$, so the network receives an image referring to the current instant and 3 past images. The temporal difference τ between the images was 300 milliseconds. Figure 3 shows a sequential robot camera image and Fig. 4 shows the images as they are in the neural network input layer.

The training phase was performed in an Intel i7-7700HQ 2.8GHz computer, 32GB DDR4 2133MHZ of RAM memory, 480GB of SSD, NVIDIA GeForce GTX 1060 6GB DDR5, running Linux Ubuntu 16.04. The programming languages used were C++ and Python; DRL models were implemented using TensorFlow,[5] based on Matthias Plappert's source-code [39].

5.2 Learning to Act as a Goalkeeper

The aim of this experiment is to investigate whether a humanoid robot, using the proposed DRL, can learn a goalkeeper role.

An episode begins with the goalkeeper robot stopped in the goal center, as shown in Fig. 5. The robot goalkeeper must remain positioned in the goal area and check if there are some ball around. Thus, as soon as the ball is detected nearby, the robot must walk towards this ball, kick it away

[2] http://www.cyberbotics.com

[3] https://www.robocuphumanoid.org/materials/rules/

[4] http://www.intel.com/content/www/us/en/nuc/overview.html

[5] https://www.tensorflow.org

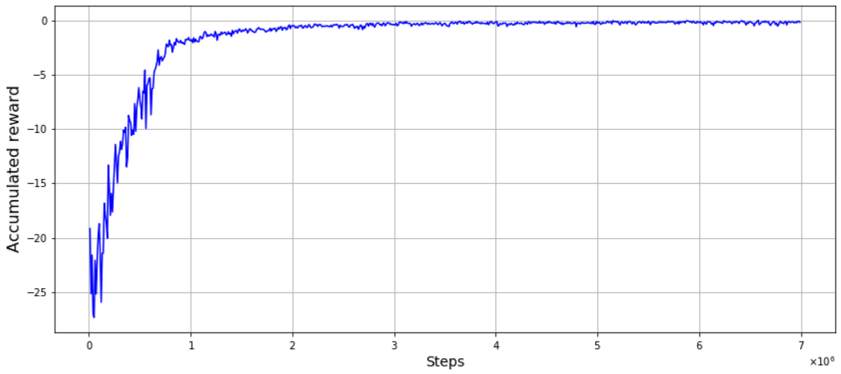

Fig. 10 Accumulated Reward

from the goal and return to the goal area. The goalkeeper robot must not jump or fall; consequently, it does not use its hands. There are no opponent robots in this experiment either.

The value of ϵ-greedy has been linearly decremented from 1.0 to 0.1 in the first 1 million steps, then remains 0.1. The complete learning process consisted of 5 million steps. At each step, a frame of the robot camera image is read and, together with the last 3 frames, forms the input of the Dueling Double DQN. The memory size $\mathcal{D}$ was 500000 steps and it always contains the most current frames, following the FIFO (First In-First Out) rule.

The rewards range from -1 to 1 and it was given as follows:

- 1.0: when the robot kicks the ball forward or to the side, and when the robot returns to the goal after kicking the ball;
- 0.5: when the Euclidean distance between the robot and the ball becomes smaller than the previous state, and when the Euclidean distance between the robot and the goal becomes smaller than the previous state (after kicked the ball);
- -0.1: when the robot stay in the same position;
- -0.5: when the robot is facing the goal; not the field as expected;
- -1.0: to all other situations.

This experiment was made considering the robot can perform 7 actions: *turn left*; *turn right*; *walk forward*; *kick right* (kick with the right leg); *kick left* (kick with the left leg); *walk backward*; *do nothing*.

The graph in Fig. 6 shows the progress of the learning. The graph shows the average steps per episode for each 10000 steps, with 400 being the maximum number of steps per episode. The red line shows the robot learning to reach the ball, while the blue line shows the learning of the complete task. The graph of Fig. 7 shows the percentage of failures that occurred for each 10000 steps. In this experiment, the failure means that the robot did not reach the aim, the aim is to go to the ball to kick it, and go back to the goal. Figure 8 shows the accumulated reward overtime.

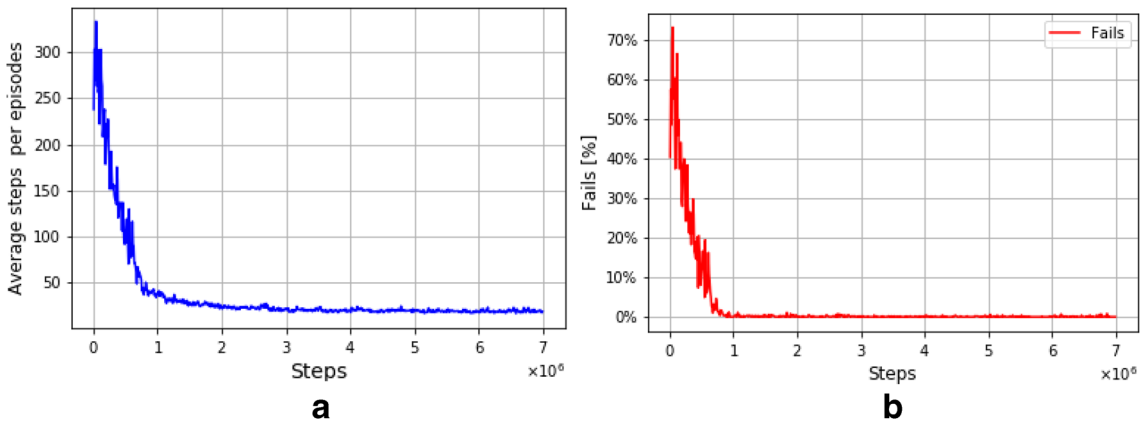

Fig. 11 Average steps and percentage of failures per 10000 steps. **a** Steps per Episode. **b** Percentage of failures per 10000 steps

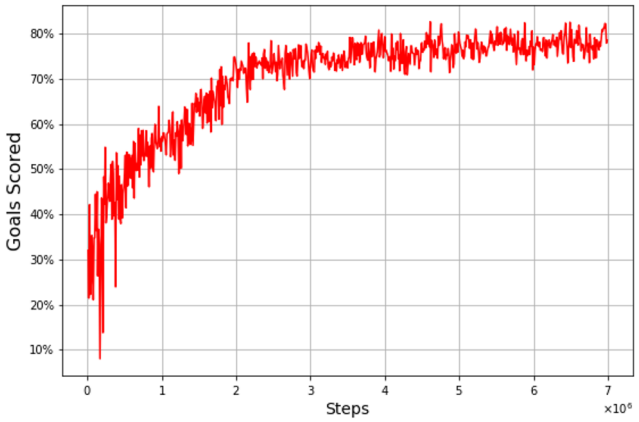

Fig. 12 Percentage of Scored Goals per 10000 steps

It is possible to notice that the Dueling Double DQN was able to maximize the rewards.

The test was performed with the network learned after the learning process. In the test, were performed 10000 episodes, where the agent failed only in 23 of 10000 episodes, with 99.77% of success.

This experiment showed that with Deep Reinforcement Learning a humanoid robot was able to learn the role of a goalkeeper robot, where the robot just kicks the ball away from the goal area and returns to the center of the goal. The experiment was performed in the Webots simulator running 5 million of steps, and the simulation was run in accelerated mode, accelerating 10 to 20 times. The source code is available at: https://github.com/Isaac25silva/DRL_goalkeeper.

5.3 Learning to Act as a Penalty Taker

The aim of this experiment is to investigate whether a robot-soccer, running our proposal, can learn to act as a penalty taker.

Figure 9 shows an example of the experiment, where the episode begins with the ball random positioned near to the penalty mark. The humanoid robot-soccer must learn to reach the ball and kick the ball through the goal, aiming to maximize the number of scored goals. The experiment was performed on the simulator, considering one penalty taker robot and no other opponent (defenders or goalkeeper).

The ϵ-greedy value has been exponentially decremented according to Eq. 4:

$$\epsilon - greedy(x) = \frac{(1-a)}{e^{(\frac{1}{b}2ex)}} + a, \tag{4}$$

where: a is the minimum exploration value; b is the number of steps to reach the minimum exploration value; x is the step the agent is on. We considered $b = 4$ million steps in order to avoid a fast decay.

The reward range from -0.1 to 1 and it was given as follows:

- 1.0: when the robot reaches the ball and kicks the ball through the goal, scoring a goal;
- -0.01: when the robot reaches the ball and kicks the ball;
- -0.03: when the Euclidean distance between the robot and the ball becomes smaller than the previous state;
- -0.05: when the robot stay in the same position;
- -0.1: to all other situations.

The robot can perform 9 actions: *turn left*; *turn right*; *walk forward*; *kick right* (kick with the right leg), *kick left* (kick with the left leg); *walk backward*; *do nothing*; *turn left around the ball*; *turn right around the ball*. These last two actions may the robot to position itself facing to the goal when it comes close to the ball.

Figure 10 shows the graph of accumulated reward overtime. It is possible to notice that the proposed DRL model was able to maximize the rewards. The graph in Fig. 11.a shows the progress of the learning, in which robot learns to reach and kick the ball. The graph show the average steps per episode for every 10000 steps, with 400 being the maximum number of steps per episode. The graph in Fig. 11.b shows the percentage of failures occurred for each 10000 steps, this is, the robot did not score the goal.

The graph in Fig. 12 shows that during the learning process, the percentage of scored goals was above 70%. The test phase, performed with 10000 episodes, the agent scored 8694 of goals of 10000 episodes, with 86.9% of success.

5.4 Transferring the Network Weights Learned on Simulator to a Real Robot

The experiment in this section aims to show that, despite some restrictions, it is possible to transfer the network weights learned in a simulator to a real-robot. This becomes interesting since performing a DRL training on a real humanoid robot is something infeasible due the amount of necessary steps to learn. We propose to perform the entire learning process (training) on a simulator and then, to transfer the network weights to the real-robot, performing the test phase. Dueling Double DQN algorithm [22] was implemented on Webots simulator and it was used to train and test the experiment of this section.

In this experiment, the robot must walk through the ball and reach it. An episode begins with the ball randomly positioned near to the robot, within the field of view of the robot and the episode ends when the robot reaches the ball or when it performs a maximum number of steps and the robot does not reach the ball.

The value of ϵ-greedy has been linearly decremented from 1.0 to 0.1 in the first 1 million steps, then remains 0.1.

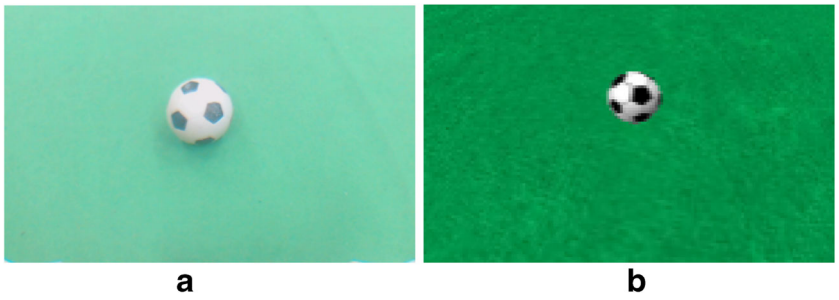

Fig. 13 Example of image captured from real robot's camera and simulated robot's camera. **a** Real Robot. **b** Simulation

The complete learning process consisted of 2.5 million steps and the memory size $\mathcal{D}$ considered was 500000 steps.

The rewards range from 1 to −1 and it was given as follows:

- 1.0: when the robot reaches the ball;
- −0.3: when the Euclidean distance between the robot and the ball becomes smaller than the previous state;
- −0.5: when the robot stay in the same position;
- −1.0 to another situations.

This experiment was made considering the robot can perform five actions: *turn left*; *turn right*; *walk forward*; *walk backward*; *do nothing*. Performing the experiment on an Intel NUC (the computer used on the humanoid robot), the inference time was about 15 milliseconds and the fastest action performed by the robot is about 300 milliseconds. Considering this, in the simulator the time 300 milliseconds was fixed to the robot perform an action.

As can be shown in Fig. 13, comparing the images captured from the robot's camera in the simulator and those captured from the real robot's camera, there are small differences that may be considered during the learning phase. For instance, the color, lightning and texture of the field, and the color and size of the ball. These kinds of variations were inserted into the simulator (in the images the robot gets) during the learning process.

It should be noted that the successful of this experiment was only possible respecting these premises:

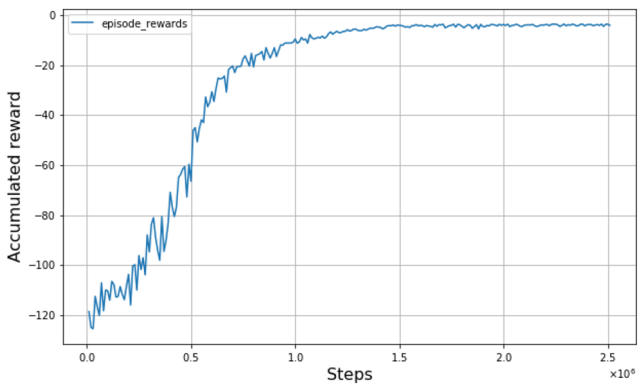

Fig. 14 Accumulated reward

- Variations on lighting, texture, and color of the field and ball: the images captured by the simulated robot's camera may have similar characteristics of images captured by the real robot's camera;
- Variations on size of ball and distortions of the camera lens: the ball of the virtual images may be distorted to look similar to the ball of real images.
- Same frame size relation: the size of frames must have the same relation in both environments (simulated and real) to avoid different distortion of the objects when performing simulated and real tests.

Figure 14 shows that the DRL algorithm was able to maximize the accumulated reward over time during the learning process. The graph in Fig. 15.a shows progress of the learning, in which robot learns to reach the ball. The graph shows the average steps per episode for every 10000 steps, and 200 being the maximum number of steps per episode. Figure 15.b shows the percentage of failures occurred for each 10000 steps, this is, the robot does not reach the ball. As can be seen, the graph decreases the percentage of failures to zero.

Aiming to test the transfer of the network weights to the real robot, the ball was positioned in three different positions, as shown in Fig. 16, and the robot should walk and reach the ball. These are: the ball positioned on the left and in front of the robot (Fig. 16a); the ball positioned in the center and in front of the robot (Fig. 16b); and the ball positioned on the right and in front of the robot (Fig. 16c).

Table 2 shows the results of performing the experiment on simulation and with real robot, with the ball positioned in three different positions. It was performed 30 episodes for each position and the results are an average and standard deviation of steps per episode performed by the robots on simulated and real environment. Two failures were observed when the ball was positioned at the left or at right and in front of the real robot. These failures occurred at the beginning of the episode. The robot performed an action that made him lose the ball in his field of view.

Figure 17 shows the steps performed by the real robot in one of the experiment samples, where the ball was positioned at the right and in front of the robot. Figure 17.a shows the beginning of the episode, Fig. 17b shows that the

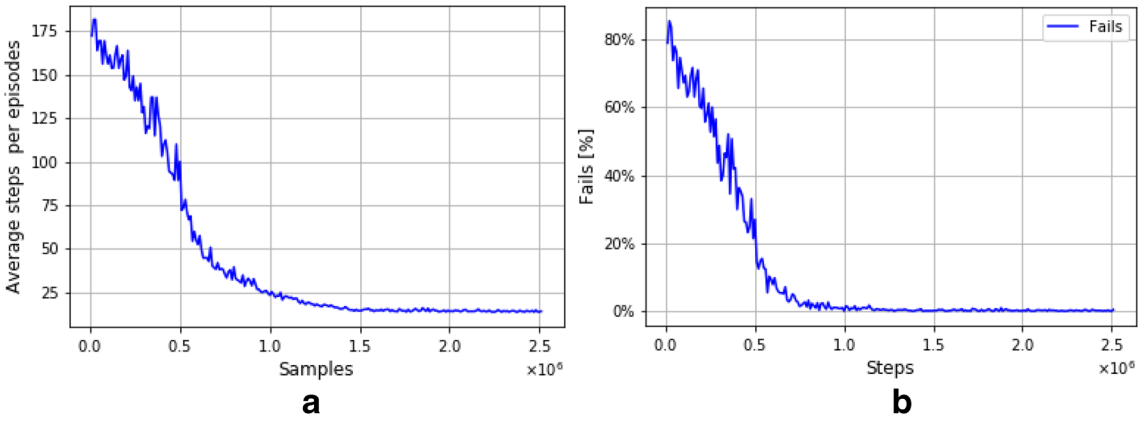

Fig. 15 Average steps and percentage of failures per 10000 steps in simulation. **a** Steps per Episode. **b** Failures per 10000 steps

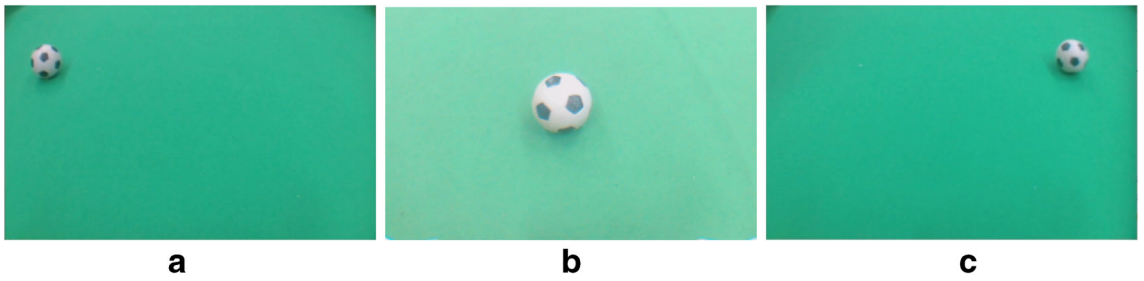

Fig. 16 Three different positions of the ball that the robot should walk and reach. **a** Left. **b** Center. **c** Right

Table 2 Steps per episode

Ball position	Real robot			Simulated		
	Mean	σ	Failures	Mean	σ	Failures
Center	15.7	7.74	0	9.90	0.88	0
Right	16.6	6.06	2	21.4	2.75	0
Left	19.2	6.65	2	18.1	2.01	0

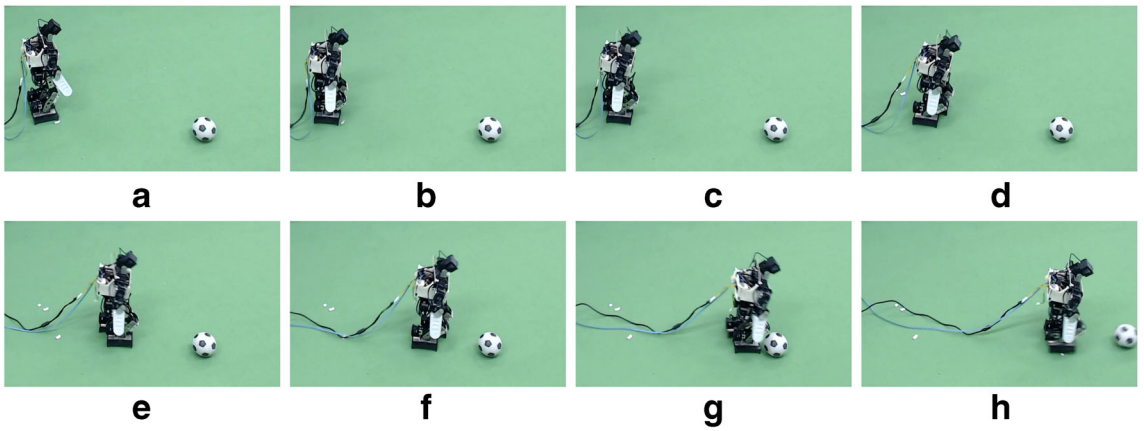

Fig. 17 Sequence of steps of the robot, where the ball was positioned at the right and in front of the robot

robot performed a turn to *right* action, Fig. 17c to Fig. 17g the robot performed actions of *walk forward*, and Fig. 17h shows that the robot reaches the ball. The video is available at https://youtu.be/GN9Qa6ydzQU.

This experiment showed that it is possible to perform the learning process in the simulator and transfer the weights of the network to a real robot. However, it is not an immediate transfer and some premises must be considered during the learning process.

The source code of this experiment is available on GitHub.[6]

5.5 Discussion

The main difference observed on experiments presented on Sections 5.2 and 5.3 is that the goalkeeper behavior learns easier than a penalty taker behavior. This is because in the goalkeeper problem the agent needs only to follow a sequence of actions to reach the goal. In the penalty taker problem, the main goal (score goals) is presented only in the last step, wherein the last step, the robot receives a reward for scoring or not the goal, and this reward must be spread to the other states. The rewards lead the robot to learn a sequence of actions to reach the ball and to position itself in a way that can score a goal.

Finally, the experiment presented on Section 5.4 differs from others while it aims to transfer the network weights learned on simulation to a real robot. This is not a trivial task and differences between virtual and real images may be considered during the training phase.

6 Conclusions

This paper shows that a robot using deep reinforcement learning algorithms is able to learn tasks by only observing the images of the environment, acquired from its own camera. The results show that the robot learned tasks related to the role of a robot-soccer player, such as goalkeeper and penalty taker. Simulation becomes necessary since the learning process requires the robot to perform and repeat the task thousands of times. As this is not feasible using a real robot, this article investigated and showed that a learned model can be reused on a real robot.

Performing the transfer of network weights learned on simulator to a real robot is possible and the real robot is able to perform the tasks as well as simulated. However, the differences of color and lighting on simulated and real images may be considered during the training phase, allowing the transfer.

One of the future works is to perform Deep Reinforcement Learning in other tasks of a soccer player robot such as: player learning when to keep possession of the ball or make a pass; attacker learning when to kick the ball into the goal. And also perform the experiment of transferring the weights from the simulator to the real robot of these players.

Acknowledgments This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Author Contributions

- Conceptualization: I. J. da Silva; D. H. Perico; T. P. D. Homem; R. A. C. Bianchi
- Methodology: I. J. da Silva; D. H. Perico; T. P. D. Homem; R. A. C. Bianchi
- Software: I. J. da Silva
- Investigation: I. J. da Silva
- Formal Analysis: I. J. da Silva; D. H. Perico; T. P. D. Homem; R. A. C. Bianchi
- Validation: I. J. da Silva; D. H. Perico; T. P. D. Homem; R. A. C. Bianchi
- Data curation: I. J. da Silva
- Writing – original draft: I. J. da Silva; D. H. Perico; T. P. D. Homem
- Writing – review & editing: I. J. da Silva; D. H. Perico; T. P. D. Homem; R. A. C. Bianchi
- Visualization: I. J. da Silva; D. H. Perico; T. P. D. Homem; R. A. C. Bianchi
- Resources: I. J. da Silva
- Funding acquisition: I. J. da Silva
- Project administration: R. A. C. Bianchi
- Supervision: R. A. C. Bianchi

Funding This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

Availability of data and materials The scripts utilized in the experiments presented in this paper are available at https://github.com/Isaac25silva/DRL_goalkeeper and https://github.com/Isaac25silva/DRLtransfer.git.

Declarations

Competing interests The authors declare that they have no conflict of interest.

References

1. Kim, S., Kim, M., Lee, J., Hwang, S., Chae, J., Park, B., Cho, H., Sim, J., Jung, J., Lee, H., et al.: Approach of team Snu to the Darpa robotics challenge finals. In: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), pp. 777–784. IEEE (2015)
2. Lim, J., Lee, I., Shim, I., Jung, H., Joe, H.M., Bae, H., Sim, O., Oh, J., Jung, T., Shin, S., et al.: Robot system of drc-hubo+ and control strategy of team kaist in darpa robotics challenge finals. Journal of Field Robotics **34**(4), 802–829 (2017)

[6] https://github.com/Isaac25silva/DRLtransfer.git

3. Jumel, F., Saraydaryan, J., Leber, R., Matignon, L., Lombardi, E., Wolf, C., Simonin, O.: Context aware robot architecture, application to the robocup@ home challenge. In: Robocup Symposium (2018)
4. Perico, D.H., Silva, I.J., Vilão, C.O. JR., Homem, T.P.D., Destro, R.C., Tonidandel, F., Bianchi, R.A.C.: Newton: a high level control humanoid robot for the robocup soccer kidsize league. In: Robotics, pp. 53–73. Springer (2014)
5. Kempka, M., Wydmuch, M., Runc, G., Toczek, J., Jaśkowski, W.: Vizdoom: a doom-based ai research platform for visual reinforcement learning. arXiv:1605.02097 (2016)
6. Lample, G., Chaplot, D.S.: Playing fps games with deep reinforcement learning. arXiv:1609.05521 (2016)
7. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. arXiv:1611.05397 (2016)
8. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. arXiv:1602.01783 (2016)
9. Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., et al.: Learning to navigate in complex environments. arXiv:1611.03673 (2016)
10. Justesen, N., Bontrager, P., Togelius, J., Risi, S.: Deep learning for video game playing. IEEE Transactions on Games (2019)
11. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. arXiv:1504.00702 (2015)
12. Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y., Levine, S.: Collective robot reinforcement learning with distributed asynchronous guided policy search. arXiv:1610.00673 (2016)
13. Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. arXiv:1610.00633 (2016)
14. Mitchell, T.M. Machine Learning, 1st edn. McGraw-Hill, Inc., New York (1997)
15. Russell, S.J., Norvig, P. Artificial Intelligence: a Modern Approach, 3rd edn. Prentice Hall, Upper Saddle River (2010)
16. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
17. Lange, S., Riedmiller, M.A.: Deep learning of visual control policies. In: ESANN. Citeseer (2010)
18. Riedmiller, M.: Neural fitted q iteration-first experiences with a data efficient neural reinforcement learning method. In: ECML, vol. 3720, pp. 317–328. Springer (2005)
19. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv:1312.5602 (2013)
20. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv:1511.05952 (2015)
21. Hasselt, H.V., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. arXiv:1509.06461 (2015)
22. Wang, Z., de Freitas, N., Lanctot, M.: Dueling network architectures for deep reinforcement learning. arXiv:1511.06581 (2016)
23. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra. D.: Continuous control with deep reinforcement learning. arXiv:1509.02971 (2015)
24. Sutton, R.S., Barto, A.G. Reinforcement Learning: an Introduction, 2nd edn. MIT Press, Cambridge (2017). in progress - draft edition
25. Lin, L.-J.: Reinforcement Learning for Robots Using Neural Networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science (1993)
26. Hasselt, H.V.: Double q-learning. In: Advances in Neural Information Processing Systems, pp. 2613–2621 (2010)
27. Tai, L., Liu, M.: Towards cognitive exploration through deep reinforcement learning for mobile robots. arXiv:1610.01733 (2016)
28. Tai, L., Li, S., Liu, M.: A deep-network solution towards model-less obstacle avoidance. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2759–2764. IEEE (2016)
29. Tai, L., Liu, M.: Mobile robots exploration through cnn-based reinforcement learning. Robotics and Biomimetics **3**(1), 24 (2016)
30. Lobos-Tsunekawa, K., Leiva, F., Ruiz-del-Solar, J.: Visual navigation for biped humanoid robots using deep reinforcement learning. IEEE Robotics and Automation Letters **3**(4), 3247–3254 (2018)
31. Abreu, M., Lau, N., Sousa, A., Reis, L.P.: Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 1–8. IEEE (2019)
32. Team description paper: Citbrains (kid size league) (2017)
33. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (2014)
34. Ha, I., Tamura, Y., Asama, H., Han, J., Hong, D.W.: Development of open humanoid platform darwin-op. In: SICE Annual Conference 2011, pp. 2178–2181 (2011)
35. Ha, I., Tamura, Y., Asama, H.: Development of open platform humanoid robot darwin-op. Adv. Robot. **27**(3), 223–232 (2013)
36. Michel, O.: Webots: professional mobile robot simulation. Journal of Advanced Robotics Systems **1**(1), 39–42 (2004)
37. Ha, I., Tamura, Y., Asama, H., Han, J., Hong, D.W.: Development of open humanoid platform darwin-op. In: SICE Annual Conference 2011, pp. 2178–2181. IEEE (2011)
38. Perico, D.H., Silva, I.J., Vilão, C.O., Homem, T.P.D., Destro, R.C., Tonidandel, F., Bianchi, R.A.C.: Hardware and software aspects of the design and assembly of a new humanoid robot for robocup soccer. In: 2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol, pp. 73–78 (2014)
39. Matthias Plappert. keras-rl. https://github.com/matthiasplappert/keras-rl (2016)

Isaac Jesus da Silva received his Bachelor's degree, Master's degree and PhD degree in Electrical Engineering from University Center FEI. He is currently an Adjunct Professor at University Center FEI. He researches in the following areas: Deep Learning applied to autonomous robots, Robot Soccer, Mobile Robots, Computer Vision and Humanoid Robots Control.

Danilo Hernani Perico received his Bachelor's degree (2009), Master's degree (2012) and PhD degree (2017) in Electrical Engineering from University Center FEI. He is currently Assistant Professor at the University Center FEI in São Bernardo do Campo, Brazil. He researches in the following areas: Robotics, Artificial Intelligence, Machine Learning, Knowledge Representation, Spatial Reasoning and Computer Vision.

Thiago Pedro Donadon Homem holds a Doctorate in Electrical Engineering from the University Center FEI (2018). He is currently Associate Professor at the Federal Institute of Education, Science and Technology of São Paulo in São Paulo, Brazil. He researches in the following areas: Robotics, Artificial Intelligence and Machine Learning.

Reinaldo Augusto da Costa Bianchi holds a Doctorate in Electrical Engineering from the University of Sao Paulo (2004). He held a internship at the Institute of Investigation en Intelligencia Artificial, Barcelona, Catalunya, Spain. He is currently Full Professor at the University Center FEI in São Bernardo do Campo, Brazil. He researches in the following areas: Robotics, Artificial Intelligence, Computer Vision, Machine Learning and Multi-Agent Systems.