Distributional Reinforcement Learning Applied to Robot Soccer Simulation

Work-In-Progress Paper - ALA Workshop at AAMAS-19

Felipe Leno Da Silva[1,2], Anna Helena Reali Costa[1], Peter Stone[2]
[1]University of São Paulo, Brazil
[2]The University of Texas at Austin, USA
{f.leno,anna.reali}@usp.br, pstone@cs.utexas.edu

ABSTRACT

Directly approximating the expected value of actions is the most traditional approach to learn in Reinforcement Learning (RL) problems. However, learning the distribution of possible returns for each action (Distributional RL) has been shown to improve learning when allied to function approximation. Although efficient Distributional RL algorithms have been proposed in recent years, its use in multiagent tasks has not been investigated so far. Moreover, Distributional RL has a yet unexplored potential to transfer learning, which makes it a promising topic for research. In this work-in-progress paper, we describe our efforts and results applying Distributional RL in *Half Field Offense*, a popular and challenging testbed for Multiagent RL. We also describe our long-term plans on how to combine Distributional RL with Multiagent Transfer Learning. To the best of our knowledge, this is the first reported use of Distributional RL in multiagent tasks. Our next step is to build upon this first effort to achieve our long-term objectives.

KEYWORDS

Transfer Learning; Reinforcement Learning; Multiagent Learning

1 INTRODUCTION

Very commonly, Reinforcement Learning (RL) agents aim at estimating an expected quality value for each state-action tuple during learning (known as Q function) [11]. However, learning a *distribution* of possible returns instead of directly computing the Q function might present advantages. Besides enabling risk-sensitive reasoning [6], learning an approximate distribution preserves multimodality and was shown to result in more effective learning than the standard approach in Atari games [2]. Inspired by the impressive results achieved by *Distributional RL* when combined with *Deep RL* techniques [5], we here apply Distributional RL to *Half Field Offense* (HFO) [3], a challenging domain that has been used as a multiagent learning testbed. We show that (i) Distributional RL is able to learn a consistent distribution of possible returns in the HFO domain; (ii) Reasoning over the distribution of returns enables learning policies comparable to the hand-tuned *helios* strategy; and (iii) *Distributional* RL achieves a better learning performance than the expected-value approach (here implemented as DQN) in two evaluated scenarios. To the best of our knowledge, this paper represents the first effort towards applying Distributional RL to a multiagent task.

2 BACKGROUND

RL is a paradigm for learning in *Markov Decision Processes* (MDP) [11]. An MDP is described by a tuple $\langle S, A, T, R \rangle$, where S is the (possibly infinite) set of states in the system, A is the set of actions available to the agent, T is the state transition function, and R is the reward function. T and R are unknown in learning problems, hence the agent must learn through samples of interactions with the environment. The agent objective is to learn a policy $\pi : S \rightarrow A$ that dictates an action to any possible state. In some of the most commonly used RL algorithms (such as Q-Learning [14]), the agent iteratively refines an estimate of the quality of each action in each state: $Q^*(s,a) = E\left[\sum_{i=0}^{\infty} \gamma^i r_i\right]$, where r_i is the reward received after i steps from using action a on state s and following the optimal policy on all subsequent steps, and γ is a discount factor. When the state-action space is discrete, the Q function can be simply represented as a table containing entries for all possible state-action pairs. However, when the state space is continuous or too big, using a function approximator (such as a Neural Network [5]) is required. An optimal policy can be extracted from Q^* as $\pi^*(s) = \arg\max_a Q^*(s,a)$. However, the Q function preserves only the expected value of the actions, while information about the *distribution* of returns is lost. Distributional RL, on the other hand, aims at improving an estimate of the distribution of possible returns, which might result in more stable learning when allied with function approximators [2]. The C51 algorithm builds a parametric distribution assigning a probability to N possible returns, evenly spaced between two parameters V_{min} and V_{max}. The goal of C51 is to learn a parametric model $\theta : S \times A \rightarrow \mathbb{R}^N$

$$Z_\theta(s,a) = z_i \text{ w.p. } p_i(s,a) = \frac{e^{\theta_i(s,a)}}{\sum_j e^{\theta_j(s,a)}}, \tag{1}$$

where Z_θ is the value sampled from the distribution, z_i is the i-th possible return value defined by the distribution parameters, $p_i(s,a)$ is the probability of Z_θ being z_i for state s and action a, and $\theta_i(s,a)$ is the value defined by a function approximator (often a Neural Network). For defining the policy, it is possible to either retrieve the Q-value from the distribution : $Q(s,a) = \sum_{i=1}^{N} z_i p_i(s,a)$ or to directly reason over the probabilities of return (we consider both possibilities in the next section). During training, the function approximator defining $\theta_i(s,a)$ is iteratively refined by minimizing the KL divergence between the learned distribution and the samples observed (for a complete description of the algorithm see [2]).

3 APPLYING DISTRIBUTIONAL RL TO HFO

Half Field Offense (HFO) is a subtask of the full RoboCup simulated soccer [4]. An offensive team tries to score a goal against a defensive team in half of the field. The episode ends if a goal is scored, the defensive team steals the ball, the ball leaves the half-field boundaries, or a time limit is reached. HFO is easily configurable and can be used to build tasks with varying levels of difficulty by changing the number of defensive or offensive agents, the starting position of the ball, and the strategy of the agents. Moreover, HFO provides high-level actions and state features based on successful teams in the RoboCup competition, as well as the possibility of quickly loading high-performance strategies for some of the agents, and composing teams of challenging opponents or high-performing teammates. In this section we perform experiments in the HFO domain and show the quality of the learned policy when using Distributional RL.

In all our experimental settings, we train the agents in the offensive team and set the defensive agents to the *Helios* strategy [1]. The C51 algorithm was implemented by building a Deep Neural network for each possible action, where each of them is composed of 3 fully-connected hidden layers with 25 neurons. The Neural Networks receive as inputs all the state variables and output values $\theta_i(s, a)$. The following state variables were used (normalized in the range $[-1, 1]$): (i) **agent position**: (X,Y) current agent position on the field; (ii) **orientation**: the global direction the agent is facing; (iii) **ball position**: (X,Y) ball current position; (iv) **proximity to the goal**: distance between the agent and the center of the goal; (v) **goal angle**: angle from the agent to the center of the goal; (vi) **goal opening**: largest open angle to the goal; (vii) **closest opponent proximity**: distance from the agent to the nearest opponent; (viii) **positions, goal opening, pass opening, and opponent proximity for each teammate.**; (ix) **position of each opponent**. When the agent has the ball possession, the available actions are: (i) **Shoot** - Tries to score a goal by shooting the ball; (ii) **Dribble** - advances the agent and ball towards the goal; (iii) **Pass** - the agent has an available *pass* action to each teammate; and (iv) **Move** - the agent moves to the best position according to the high-level action set available to standard HFO agents. In case the agent does not have the ball, only the latter action is applicable. The three first ones can be applied only when in possession of the ball. In case the agent scores a goal, a reward of +1 is given. For all unsuccessful episode terminations, the agent receives a reward of -1. A reward of 0 is given otherwise. We also evaluate the learning performance of Deep Q-Networks (DQN) [5] as a baseline for expected-value RL. All algorithms are trained by batch updates using Prioritized Experience Replay [7]. All implementations can be downloaded at https://github.com/f-leno/Distributional_HFO.

3.1 1x1 HFO

Our first experiment involves a 1x1 HFO task. A learning agent tries to score a goal against a keeper following the *Helios* strategy. When we set the *Helios* strategy to the offensive agent, it scores in roughly 82% of the attempts. Therefore, this is a task where a very high performance level can be obtained. Our main goal in this scenario is to evaluate the distribution learned by the agent, since it is easier to visualize and interpret it when only a single learning agent is present in the environment. We train the agent for 10,000 learning episodes.

Figure 1 shows the learned distribution in three states after the agent has finished training. In the left state the agent is in the initial state. Notice that, since only one offensive agent is in the field, the *pass* action is not enabled. We depict here only states where the agent has the ball possession to illustrate how the distribution influences the decision making, hence *move* is not shown in the graphs. Both *shoot* and *dribble* present a bi-modal distribution, which corresponds to the reward function that returns either -1 or $+1$ in the end of the episode. Both actions have a large probability mass around $+1$ because if the agent incorrectly shoots the ball from so far, it is still possible that the agent will reach the ball before the keeper and not lose the possession. However, since *dribble* has a larger probability mass around $+1$ than *shoot*, the agent chooses to dribble. In the state shown in the central image, the agent got closer to the goal, but not enough to score a goal with certainty. Moreover, in case of not scoring when shooting, the keeper will have the possession of the ball, terminating the episode in failure. The much larger probability mass around -1 for the *shoot* action is consistent with the agent current state. Since the *dribble* action has not changed much its distribution from the left image, the agent chooses to *dribble* in this case as well. Finally, in the right state the agent is very close to the goal and has a good opening for shooting. The *dribble* action is good enough for avoiding running over the keeper and losing the ball in most cases, hence the probability mass around $+1$ is still large. However, *shoot* has now a slightly larger probability mass around $+1$, causing the agent to shoot the ball and score the goal.

In addition to building a Q-table from the distribution (as discussed in Section 2 - hereafter named as *C51Average*), we also evaluate reasoning over the distribution when the agent has the ball possession as follows (hereafter named as *C51Threshold*):

$$\pi(s) = \begin{cases} Shoot & \text{if } p_{z>0}(s, Shoot) > 0.5 \\ Pass & \text{else if } p_{z>0}(s, Pass) > 0.5 \\ Dribble & \text{otherwise} \end{cases} \tag{2}$$

where $p_{z>0}(s, a)$ is the probability of achieving a return greater than 0 when applying action a in state s according to the learned distribution. This policy was devised following the intuition that dribbling is safe in most situations, while mistimed shots almost always cause the episode to end negatively and misplaced passes make the agents lose a lot of time. We observed in preliminary experiments that the policy of expected value-based RL would very frequently cause the agents to only pass the ball to each other or to shoot as soon as in possession of the ball because the *dribble* action had a slightly lower value during training. Hence, in this way, the agents would only shoot or pass when reasoning that there is a good probability of scoring a goal after applying those actions. The thresholds were not overly optimized, as an approach very sensitive to hyperparameters is less likely to be effective in different scenarios, which would defeat our purpose in this paper.

Figure 2 shows the performance progression during the training process. Although DQN is capable of learning an effective policy that scores around 60% of the attempts, both Distributional RL algorithms achieve a high performance faster. *C51Average* also converges to a better performance than *DQN*, scoring 80% of the

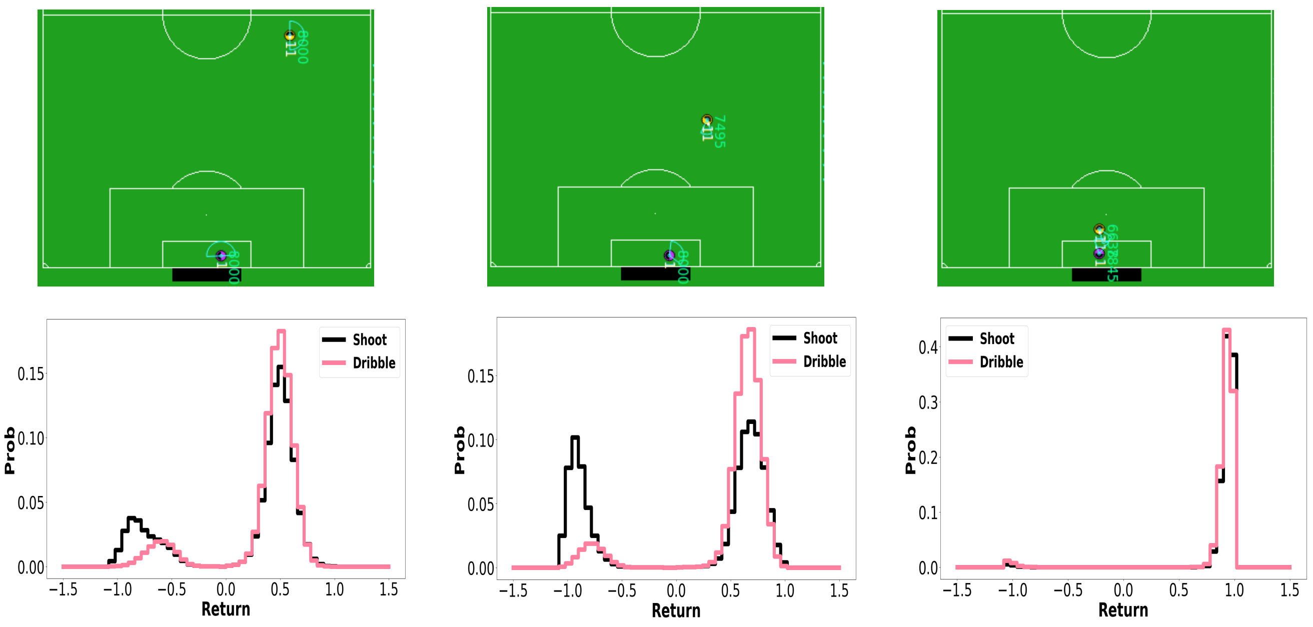

Figure 1: States in the 1x1 scenario (top - point of view of the goalkeeper) and their respective learned distributions of values for each action (bottom).

attempts after roughly 1, 700 learning steps, and eventually achieving a slightly better performance the handtuned *Helios* strategy. *C51Threshold* learns to score 60% of attempts very quickly, taking advantage of the domain-specific probabilistic reasoning. However, it eventually converges to roughly the same performance as *DQN*, showing that it would require further tuning on the thresholds to reach the same performance as *C51Average*.

3.2 2x2 HFO

We here evaluate a more complex learning problem. In addition to the goalkeeper, the defense team now has a defensive field player. The offensive team now is composed of two learning agents trying to score the goal, which means that now the *Pass* action is enabled. Learning in this scenario is much more challenging. The presence of an additional defensive player reduces drastically the probability of scoring a goal randomly, and badly-timed passes might be intercepted or make the agents waste time to recover the ball. When using the *Helios* strategy, the offensive team scores in roughly 60% of the attempts.

Figure 3 reflects the difficulty of learning in this scenario. Both DQN and *C51Average* fail to learn how to score goals consistently. Since applying *Shoot* or *Pass* actions at the wrong time most likely results in a failed episode, those algorithms are not able to get enough positive examples by using ϵ-greedy over the Q-function. By the end of the training process, the DQN policy is basically passing between the 2 agents with no clear strategic purpose and almost never shoots. We believe that this behavior was acquired after the agents observed that passing when having a clear passing angle usually does not terminate the episode, while when mistakenly shooting the episode would terminate immediately in failure. The agents were not able to consistently identify good opportunities to shoot yet after the 10, 000 learning steps. *C51Average* agents alternate between passing the ball and occasionally dribbling, both of them usually advancing towards the goal. When in very open positions the agents are able to shoot the ball and score a goal. However, they are not very consistent in achieving an open position, and they seem to achieve it only by luck. Some misplaced shots are also occasionally executed, resulting in a very bad performance overall.

However, *C51Threshold* was able to score roughly 40% of attempts by the end of training. Due to the domain-specific probabilistic reasoning the agent performed a more risk-sensitive exploratory behavior, avoiding to pass and shoot in states where those actions failed before. By the end of training, the agents have developed a behavior that is effective in some situations. Either the agents try to score individually by carrying the ball and shooting when having an open goal angle, or they pass the ball when they expect their teammate will be able to score. A curious behavior has been repeated by the agents with good effectiveness. When marked by the opponent field agent close to the goal, the agent with the ball would pass back to the other agent that is usually behind, this agent will then dribble for a short amount of steps and attempt a shoot. This behavior was mostly effective in the observed evaluations and resulted in scored goals frequently. The agents applying *C51Threshold* are much better in evaluating when to shoot the ball, scoring most of the time when shooting. Although already achieving a good performance, by the end of the 10, 000 steps the agents are much better in some portions of the field. Therefore we expect the performance could be increased by training the agents for a longer time.

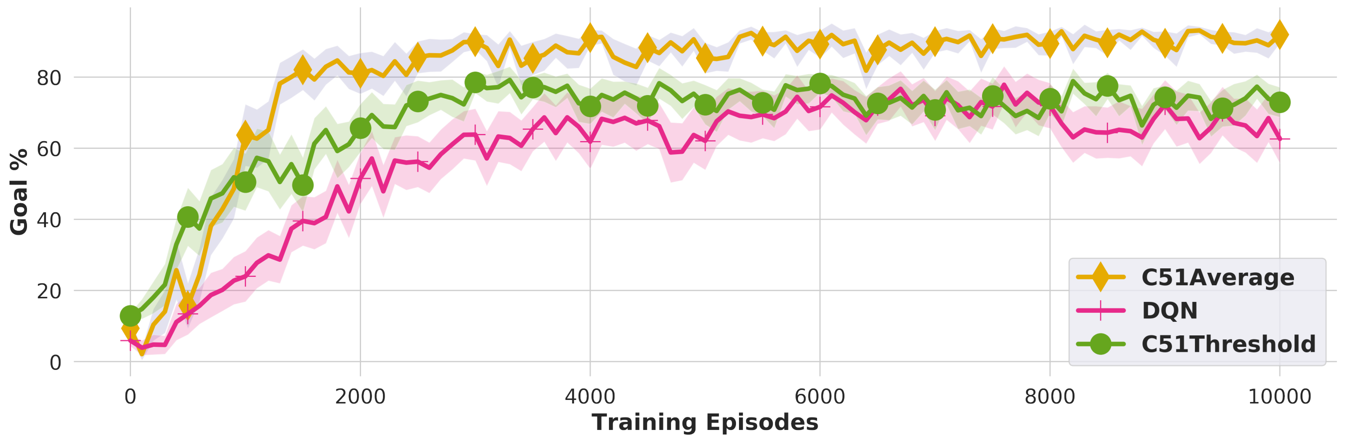

Figure 2: Average of scored goals in 100 **evaluation episodes after every** 100 **learning episodes in the 1x1 scenario. The shaded area corresponds to the** 95% **confidence interval observed in** 50 **repetitions.**

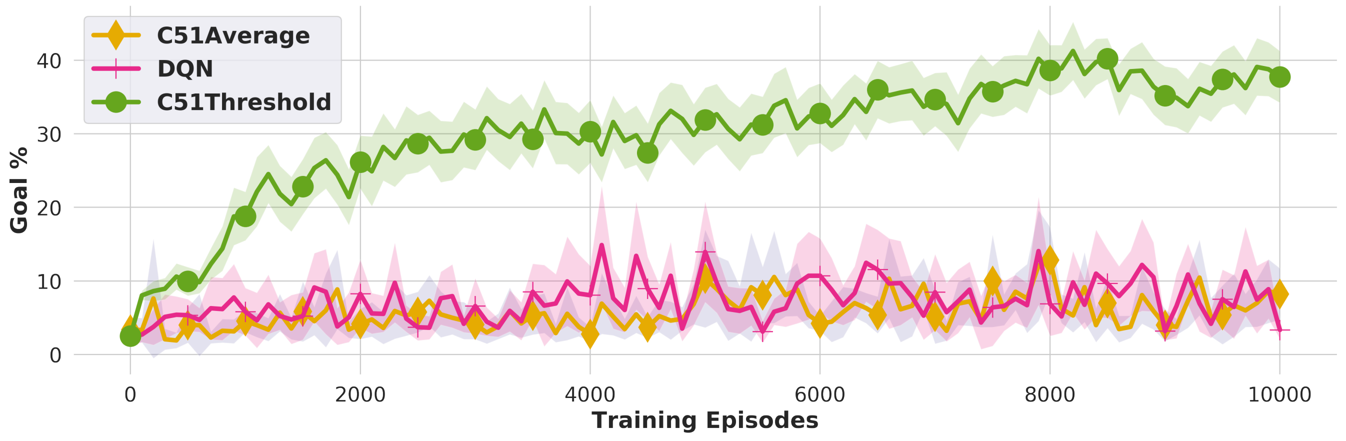

Figure 3: Average of scored goals in 100 **evaluation episodes after every** 100 **learning episodes in the 2x2 scenario. The shaded area corresponds to the** 95% **confidence interval observed in** 50 **repetitions.**

Both experimental scenarios indicate that Distributional RL performs well in HFO, in agreement with the findings of Bellemare *et al.* [2] in the Atari domain. Even in the second scenario where DQN failed to learn in the given training time, *C51Threshold* leveraged the probabilistic reasoning enabled by Distributional RL to learn a good policy.

4 WHY USE DISTRIBUTIONAL RL IN MULTIAGENT SYSTEMS?

Apart from the improvements shown in the last section, having the distribution of returns available instead of only the averages might present other advantages that we will explore in further work. The use of Distributional RL for Transfer Learning [10, 13] remains unexplored. The *Ad Hoc Advising* framework [9] enables the exchange of knowledge between agents through action advising. However, the agents need an estimate of their confidence on the current policy, which is not easy to define when the policy is extracted directly from Q-values.

However, if the agents use Distributional RL, the distribution of returns is more informative about how uncertain the agent is in the current policy. Therefore a combination of *Ad Hoc Advising* with Distributional RL is a very promising avenue. We further discuss this line in Section 5.1.

Distributional RL has also been used to achieve sensitiveness to risk. Serrano-Cuevas *et al.* [8] propose a risk-sensitive method that plans over the learned distribution of values. Risk-sensitive methods based on Distributional RL could also be extended to the multiagent case.

5 CONCLUSION AND FURTHER WORK

In this paper we evaluated the use of Distributional RL in the HFO domain. To the best of our knowledge this is the first reported use of Distributional RL in a multiagent task, and opened new research opportunities in multiagent RL. The C51 Distributional RL algorithm achieved a better performance than DQN in two evaluated scenarios, and enabled us to define policies by reasoning over the distribution of possible returns.

Our next research direction will be build upon the experiments here reported to perform Transfer Learning experiments, leveraging Distributional RL to extract a confidence measure on the current policy that will be useful for the *Ad Hoc Advising* transfer framework.

5.1 Integrating Distributional RL and Ad Hoc Advising

We here briefly discuss our initial thoughts on how the next steps in our research could be carried out.

The *Ad Hoc Advising* framework [9] intends to accelerate learning in Multiagent RL by knowledge exchange between a group of agents that are simultaneously learning in a shared environment. Each agent has a confidence function $\Upsilon : S \rightarrow \mathbb{R}$, from which the agent is able to estimate its confidence on its own policy for a given state. At every learning step, the agent evaluates $\Upsilon(s)$ for the current state s. If the confidence is low, a request for help is broadcasted to the other agents in the system, which, if they are confident enough in their own policies, might answer with an action suggestion that the *advisee* might follow.

Although this framework was able to accelerate learning in the HFO domain, the original publication defines $\Upsilon(s)$ as follows:

$$\Upsilon_{visit}(s) = \sqrt{n_{visits}(s)}, \tag{3}$$

where n_{visits} is the number of times the agent visited a particular state. The intuition of this function is that, as higher the number of visits in a state, as better will be the agent performance. This does not hold in all cases but achieved a good performance for the advising framework. However, this function has two main inconveniences: (i) the number of visits for all states must be stored; and (ii) it is inapplicable if the exact same state is not expected to be visited many times.

The original publication applied the framework to HFO by discretizing the state space through tile code. However, in addition of having to store a table of visit counts for each discretized state, tile coding is also not applicable in every task (e.g., tasks in each the state is observed through an image). Although previous versions of the framework used the Q-table to derive confidence functions [12], the Q-values are not themselves enough to approximate a confidence on the state until the advisor converged to an optimal policy.

However, the Distributional RL might enable us to derive a confidence function directly from the learned distribution of values. The main research question to be answered next is how to do it. Since the agents start learning from a roughly uniform distribution and then specialize it to the task, the shape of the distribution might be usable as a proxy of for how long the agent has been training. Evaluating the entropy of the distribution seems to be a promising starting point.

ACKNOWLEDGEMENTS

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-1637736, IIS-1651089, IIS-1724157), Intel, Raytheon, and Lockheed Martin. Peter Stone serves on the Board of Directors of Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research. We also gratefully acknowledge financial support from CNPq, grants 425860/2016-7 and 307027/2017-1 and São Paulo Research Foundation (FAPESP), grants 2015/16310-4, 2016/21047-3, and 2018/00344-5.

REFERENCES

[1] Hidehisa Akiyama. 2012. Helios team base code. https://osdn.jp/projects/rctools/. (2012).

[2] Marc G. Bellemare, Will Dabney, and Rémi Munos. 2017. A Distributional Perspective on Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML).*

[3] Matthew Hausknecht, Prannoy Mupparaju, Sandeep Subramanian, Shivaram Kalyanakrishnan, and Peter Stone. 2016. Half Field Offense: An Environment for Multiagent Learning and Ad Hoc Teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop.*

[4] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. 1997. RoboCup: A Challenge Problem for AI. *AI magazine* 18, 1 (1997), 73.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level Control through Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529–533. https://doi.org/10.1038/nature14236

[6] Tetsuro Morimura, Masashi Sugiyama, Hisashi Kashima, Hirotaka Hachiya, and Toshiyuki Tanaka. 2010. Parametric Return Density Estimation for Reinforcement Learning. *arXiv preprint arXiv:1203.3497* (2010).

[7] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized Experience Replay. *CoRR* abs/1511.05952 (2015). arXiv:1511.05952 http://arxiv.org/abs/1511.05952

[8] Jonathan Serrano-Cuevas, Eduardo F Morales, Pablo Hernandez-Leal, Daan Bloembergen, and Michael Kaisers. 2018. Learning on a Budget Using Distributional RL. In *AAMAS Adaptive Learning Agents (ALA) Workshop.*

[9] Felipe Leno Da Silva, Ruben Glatt, and Anna Helena Reali Costa. 2017. Simultaneously Learning and Advising in Multiagent Reinforcement Learning. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS).* 1100–1108.

[10] Felipe Leno Da Silva, Matthew E. Taylor, and Anna Helena Reali Costa. 2018. Autonomously Reusing Knowledge in Multiagent Reinforcement Learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI).* 5487–5493.

[11] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction* (1st ed.). MIT Press, Cambridge, MA, USA.

[12] Matthew E. Taylor, Nicholas Carboni, Anestis Fachantidis, Ioannis P. Vlahavas, and Lisa Torrey. 2014. Reinforcement Learning Agents Providing Advice in Complex Video Games. *Connection Science* 26, 1 (2014), 45–63. https://doi.org/10.1080/09540091.2014.885279

[13] Matthew E. Taylor and Peter Stone. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research (JMLR)* 10 (2009), 1633–1685. https://doi.org/10.1145/1577069.1755839

[14] Christopher J. Watkins and Peter Dayan. 1992. Q-Learning. *Machine Learning* 8, 3 (1992), 279–292.