Learning to Play RoboCup Soccer from Scratch using Deep Learning

Ofir Zafrir*, Tomer Schweppe*, Daniel J. Mankowitz, Tom Zahavy
Electrical Engineering Department
The Technion - Israel Institute of Technology
Haifa 32000, Israel

Abstract

The novel approach of Deep Deterministic Policy Gradient (DDPG) presented in [Lillicrap et al. (2015)] has shown great success in tackling Reinforcement Learning (RL) problems featuring continuous state and action spaces. [Hausknecht and Stone (2015)] extended the DDPG algorithm for solving multiple parameterized continuous action space implemented in the Half-Field-Offense (HFO) RoboCup environment. In this paper we propose an alternative architecture to solve the HFO RL problem in terms of network size and training time while preserving state of the art performance. We also present an implementation of skills method which show great promise towards solving multiple agents problems in RoboCup environment but is not limited to RoboCup. To the extent of our knowledge the skills method wasn't implemented before in a continuous state and multiple parameterized continuous action space.

Introduction

RoboCup is an initiative started in 1995 with the purpose to present a standard problem in the form of soccer that will accelerate AI, Robotics and Machine Learning research. Since RoboCup presents a vast number of problems need solving in order to perform a soccer game from creating an autonomous agent, multi-agent collaboration to sensor-fusion and robotics, RoboCup offers a software environment for research on the software aspects of RoboCup [Kitano et al. (1995)].

The Half Field Offense (HFO) platform is one of the RoboCup software platforms. The HFO platform is a solution for experimenting and testing algorithms which is easy to setup and use ridding the user from the cumbersome work of setting up the environment which can take weeks otherwise. The HFO platform offers an API for controlling the agent, retrieving the state of the environment and bench-marking the agent [Hausknecht, Mupparaju, and Subramanian].

OpenAI Gym is a toolkit for Reinforcement Learning (RL) research and bench-marking designed by OpenAI. OpenAI Gym offers a single python API to many RL problems, moreover OpenAI Gym lets it's users upload their benchmark to a shared website where people can compare the performance of algorithms. One of the environment OpenAI Gym offers is Gym-soccer. Gym-soccer breaks down the game of soccer into sub problems and offers to solve them using RL[Brockman et al. (2016)].

In this paper we present our attempts to solve the sub problem of learning a policy for an agent to score a goal against a keeper. To solve the problem we use the low level API of HFO which consist of continuous state space and continuous action space. The continuous space is a challenge in the world of RL. An algorithm called Deep Q Networks (DQN) [Mnih et al. (2015)] was shown to solve RL problems with great success. DQN handles the large state space very well but it is designed to only handle small discrete action space. To solve the problem of continuous action space an algorithm called Deep Deterministic Policy Gradient (DDPG) [Lillicrap et al. (2015)] which combines an actor-critic architecture with DQN. DDPG was shown to solve many problems with continuous action space. Later an extension of the DDPG algorithm was proposed to learn a policy in a bounded continuous action space using inverting gradients [Hausknecht and Stone (2015)]. It is shown that the extended DDPG algorithm solves the problem of learning a policy for an agent to score against an empty goal, but the algorithms fails at the task of scoring against a keeper which leads to the conclusion that solving the entire problem at once is too complex for the exisiting methods and therefore a new approache is required. The key components of our approach is dividing the field into areas where different policies are required, for each area of the field we assign a skill to direct the policy in that area. For training we use the DDPG algorithm with an extension of skills we implemented.

There have been other works in the field of RoboCup that attempted to solve a similar problem, for example: [Cohen et al.] used images as the state space instead the one offered by HFO and used the higer level API which is not continuous. [Hausknecht and Stone (2015)] has shown state of the art results scoring against an empty goal but failed to score against a keeper. [Barrett and Stone (2015)] has

*These authors contributed equally

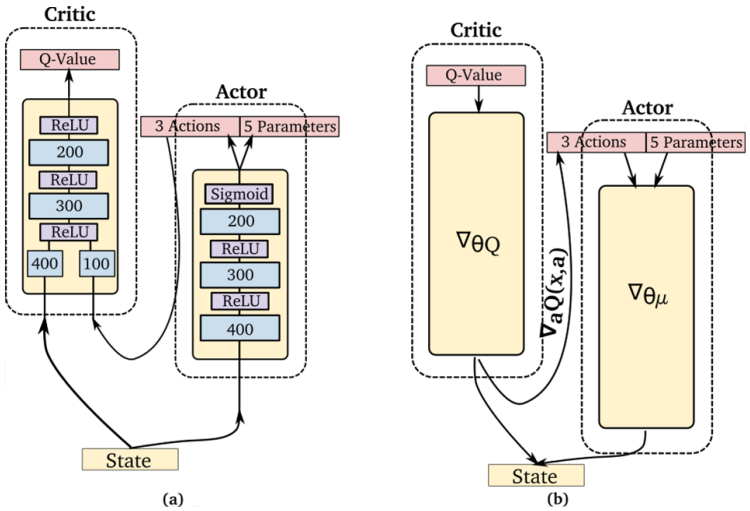

Figure 1: **Actor-Critic architecture (a):** the state is fed to the actor network producing an action. The action than is fed to the critic together with the state. The state and action are concatenated at the second layer of the critic network. **Actor update (b):** In the backwards pass we generate the critic gradients w.r.t the action and we back-propagate them to the actor network and update the actor network. We ignore the critic gradients w.r.t critic's parameters.

shown success in achieving teamwork and scoring against a keeper but also used the high level API. [Mankowitz, Mann, and Mannor (2016)] has shown the use of skills in the HFO environment using ASAP method offered in the article and have succeeded in scoring against a keeper but also used the high level API which is not continuous.

Background

Reinforcement Learning (RL) is a domain in machine learning. A standard RL problem consists of an environment and an agent. In each time step the environment produces a state S, the agent given the state takes an action A on the environment which returns a reward $r(s_t, a_t)$ and a new state S'. The goal is to find a policy $\pi : S \rightarrow A$ for the agent that will maximize the accumulated reward over time $G_t = \sum_{k=t}^{\infty} \gamma^k r_(s_{k+1}, a_{k+1})$ where $\gamma \in [0, 1]$ is the discount factor which decides the present value of the future rewards.

The action-value function describes the expected return starting from state s, taking action a and then following policy π:

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi[G_t|s_t, a_t] \tag{1}$$

We also define of the recursive relationship known as the Bellman equation for the action-value function:

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})|s_t, s_t] \tag{2}$$

Since the target policy is deterministic we avoided taking the expectation on the inner Q. Both equations are used in many RL algorithms.

Q-Learning [Watkins and Dayan (1992)] is an off-policy algorithm which uses the greedy policy $\mu(s) = \arg\max_{a'} Q(S_{t+1}, a')$. We define our loss with the Temporal Difference (TD) error:

$$L(\theta^Q) = \mathbb{E}[(Q(s_t, a_t|\theta^Q) - y_t)^2] \tag{3}$$

where

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q) \tag{4}$$

Where θ^Q represents the parameters of the function we use to approximate Q which we want to optimize by minimizing the loss. We also notice that y_t is also dependent on θ^Q.

Deep Q Networks (DQN) is an adaptation of the Q-Learning algorithm [Mnih et al. (2015)]. Since Q-Learning as is wasn't stable, DQN has made the following 2 major changes: use of a replay buffer storing the experience of the agent behavior in order to learn from it, and a separate target network parameterized by $\theta^{Q'}$ to calculate y_t where $\theta^{Q'}$ follows θ^Q with a slow update rate. Those two major changes allowed DQN to achieve super human level control in a number of Atari games.

Deep Deterministic Policy Gradient (DDPG) [Lillicrap et al. (2015)] offers a solution to applying DQN to continuous actions spaces. Since we have a continuous action space it is very difficult to apply Q-Learning as is. Since Q-Learning uses a greedy policy which finds at each step $\mu(s_t)$ which includes an optimization of a_t. It is a very slow and not practical to do this optimization. DDPG uses an actor-critic approach based on the DPG algorithm [Silver et al. (2014)]. In this approach the policy is a parameterized function $\mu(s|\theta^\mu)$. The critic $Q(s, a)$ is learned like in DQN by minimizing the loss:

$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \tag{5}$$

where Q', μ' are the target critic and action networks like in the DQN algorithm and:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) \tag{6}$$

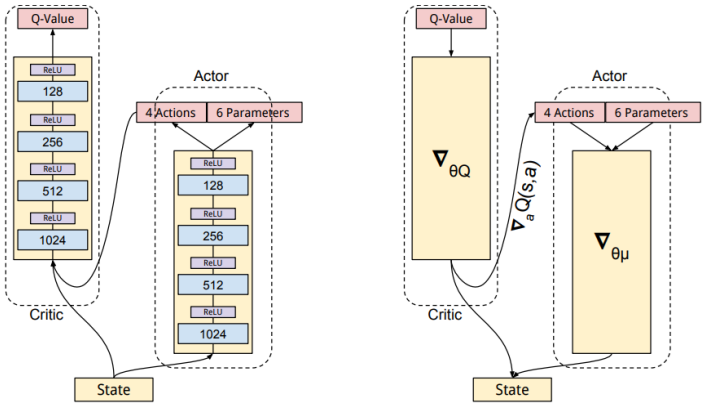

Figure 2: The architecture offered by [Hausknecht and Stone (2015)] with Inverted Gradients method. The update concept is the same as in Figure 1.

The actor is updated using the policy sampled gradient:

$$\nabla_{\theta^\mu}\mu = \frac{1}{N}\sum_i \nabla_a Q(s_i, \mu(s_i)|\theta^Q)\nabla_{\theta^\mu}\mu(s_i|\theta^\mu) \qquad (7)$$

Method

Our network architecture is an extension of the DDPG architecture suggested by [Lillicrap et al. (2015)] as shown in Figure 1. Both the actor and critic employ the same architecture. The state inputs are processed by 3 fully connected layers consisting of 400-300-200 units respectively, we have 58 or 66 input in the empty goal and keeper problem respectively. Each fully connected layer is followed by a LeakyReLU activation with negative slope of 10^{-2} except the last layers in both networks. The final layer in the actor network is connected to a sigmoid activation giving a vector of 8 values. The first 3 values are used to determine which action to execute dash, turn and kick, the action chosen has the maximum value of the 3. The next 5 values are the parameters for the action, each action has a power and angle parameter except the turn action which doesn't have a power parameter. The final layer of the critic is not followed by an activation. Weights are initialized with a truncated normal distribution with a standard deviation 10^{-2}. In addition to the state inputs, the critic network also takes as input the output of the action layer. We used the ADAM optimizer for both actor and critic. The learning rate of the actor is 10^{-4} and the learning rate of the critic is 10^{-3} and decaying through out the training process. Target networks track the actor and critic using $\tau = 10^{-4}$. In the skills architecture the actor has 2 parallel output layers, each having it's own ADAM optimizer. The condition of which skill to use is hard coded by the developer. The critic receives only the output of the chosen skill therefore the skill that was not chosen is not updated when not chosen.

The major changes are another hidden layer in both the networks in the state path and another hidden layer in the action path in the critic network, we used sigmoid activation instead of tanh, gradual learning rate and skills. We also experimented with the architecture suggested by [Hausknecht and Stone (2015)] shown in Figure 2 which uses the inverting gradient method.

Experiments

All our agents were trained for 50,000 episodes, each episode lasted for maximum number of 500 frames. On each frame we performed a learning iteration on a mini-batch of 32 samples from the replay buffer. Training each agent to approximately one day on a an Intel Skylake core i7-6700 CPU.

Our first experiment was based on the work of [Hausknecht and Stone (2015)] there we wanted to reproduce the results against an empty goal using the same architecture shown in Figure 2. In this experiment we weren't able to converge on policy that even manage to approach the ball. We have tried several sets of hyper-parameters with this approach to no avail.

In order to reproduce the results against an empty goal we based our new architecture on the one that was offered in [Lillicrap et al. (2015)]. With this architecture we managed to reproduce the results against an empty goal and managed to receive a scoring rate of 98.8%.

Our next attempt was to try our architecture against a keeper. In this setup the agent learned to approach the ball and sometimes kick it but the policy of the agent converged, the agent just approached the ball and moved it around a bit.

Through out our experiments we noticed that different approaches are needed so we tried the skills approach we described by dividing the environment in half. On the first half the skill the agent needs to learn is to approach the ball and dribble it towards the goal and on the second half the skill the agent needs to learn is to shoot the ball towards the goal. We first examined this approach against an empty goal. The scoring rate of the agent with skills against an empty goal is 95.7% which is less than the agent without skills, however we observed that the agent dribbled the ball towards the goal instead of kicking it in full power towards

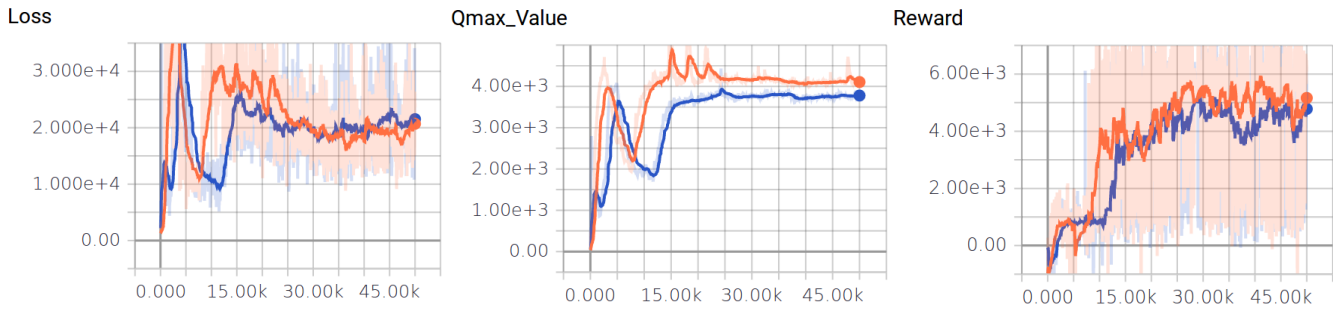

Figure 3: **Empty Goal training:** The graphs above show the performance of training the agent as a function of the number of episodes: Average Loss, Max Q-Value And Accumulated Reward. The orange plot represents the agent without skills while the blue plot represents the agent with skills. We note that while the performance of the agent without skills is better the behavior of the agent with skills incorporates dribbling

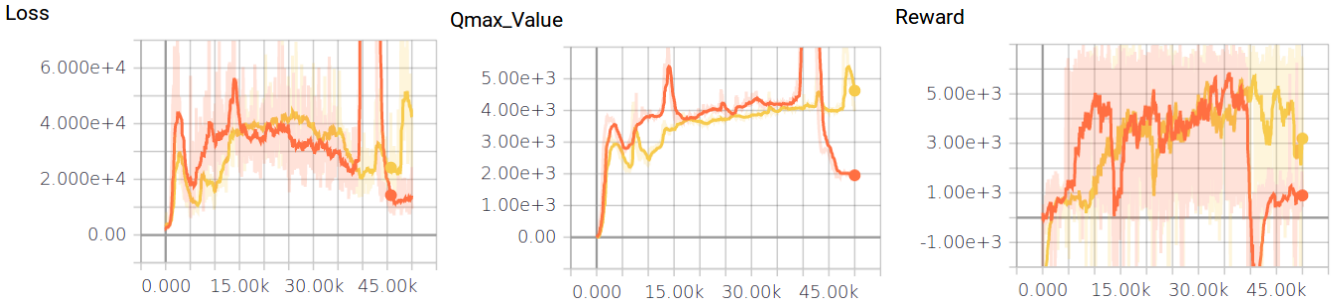

Figure 4: **Static Keeper training:** The graphs above show the same statistics of the agent as in Figure 3. The agent here is playing against a static keeper. The orange plot represents the agent without skills while the yellow plot represents the agent with skills. We note that the agent with skills shows more stability during training.

the goal which is a desired behavior in order to surpass a goalkeeper. A comparison of the two agents can be seen in Figure 3

We observed that the keeper is really good against the agent at early steps so we attempted to do gradual learning of the agent. We have designed an experiment where the agent learns to score against a static keeper, the keeper at the beginning of each episode was placed in the middle of the goal and didn't move. We have trained both non-skills and skills model against the static keeper and the results of the experiment are presented in Figure 4. At test time both agents achieved a scoring rate of 95% and performance were very similar except that the agent with skills learned to dribbled the ball where the agent without skills just kicked it as far as it can towards the goal. We also note that the addition of the static keeper didn't harm the scoring rate of the skills agent where the non-skills agent scoring rate reduced by 5%. Also at training time it is visible in Figure 4 that the non-skills progress wasn't stable in comparison the skills agent.

Discussion

This paper presents a different architecture than [Hausknecht and Stone (2015)] for solving a RL problem, learning to score against an empty goal in Robocup HFO environment from scratch. The advantage of our architecture is that it has fewer learned parameters by factor of 3.5. As a result our convergence time is much shorter, 24 hours using Intel Skylake core i7-6700 CPU versus 72 hours using NVidia Titan-X GPU. Furthermore we demonstrated that [Hausknecht and Stone (2015)] method of Inverting Gradients is not essential for solving the Empty Goal problem. Furthermore our agent scores goals more reliably than a hand-coded expert policy and is as good as [Hausknecht and Stone (2015)] learned agent.

Our work further shows the strength of the DDPG Algorithm [Lillicrap et al. (2015)] in solving RL problems from scratch in a continuous state and multiple continuous parameterized action space.

Our innovative contribution is implementing the skills architecture as described in Method section. The agent trained with skills architecture demonstrated unique behavior, dribbling - instead of kicking the ball with maximum power towards the goal until it scored, the agent dribbled the ball towards the goal until it was near the goal and scored with one powerful kick. Dribbling is a desirable behavior for playing soccer in a multiple agents environment. We assume this unique behavior is a results of the use of a separate optimizer for each skill that acts to maximize it's reward on his own working area. We noticed that if the agent prolong his training his policy converges on the two kick policy which we believe is the most rewarding policy in this problem.

We have experimented with several methods as de-

scribed in the Methods section for scoring against a keeper. In our experiments against a static keeper we have achieved with both models good performance. We noticed the skills agent kept its performance and dribbling behavior with the addition of a static keeper and in contrast the non-skills agent experienced a reduction of 5% in performance which strengthens the notion that skills are needed in order to solve the problem of scoring against a keeper. For future work we propose an adversarial approach where both the offensive player and the keeper are trained simultaneously. We also propose to learn the skills configuration instead of hand crafting it, devise an algorithm for learning the number of skills needed and the working area of each skill.

References

Barrett, S., and Stone, P. 2015. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In *AAAI*, 2010–2016.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Cohen, Y.; Krupnik, O.; Mankowitz, D. J.; Zahavy, T.; and Mannor, S. Deep reinforcement learning in robocup 2d soccer.

Hausknecht, M., and Stone, P. 2015. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*.

Hausknecht, M.; Mupparaju, P.; and Subramanian, S. Half field offense: An environment for multiagent learning and ad hoc teamwork.

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; and Osawa, E. 1995. Robocup: The robot world cup initiative.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mankowitz, D. J.; Mann, T. A.; and Mannor, S. 2016. Adaptive skills adaptive partitions (asap). In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 1588–1596.

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. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *ICML*.

Watkins, C. J., and Dayan, P. 1992. Q-learning. *Machine learning* 8(3-4):279–292.