Deep Reinforcement Learning for Pendubot

Theo Vincent[1], **Boris Belousov**[1]
[1]German Research Center for AI (DFKI), Research Department: Systems AI for Robot Learning
{name.surname}@dfki.de

Abstract

DQN is classically known as a good controller for complex stochastic tasks with discrete action spaces. In this report, we investigate its ability to solve the Pendulum challenge pointing out some intuition on how to generalize this algorithm to continuous action spaces.

1 Introduction

Reinforcement Learning has been shown to be effective in solving a wide variety of tasks [1; 2]. We use Deep-Q Network (DQN) [3] to overcome classical control approach limitations for this challenge [4]. We manage to make DQN learn a hand-crafted reward that leads to an interesting behavior.

2 Method

For this challenge, we used DQN to solve the Pendubot task [5]. DQN uses the theory of Reinforcement Learning. The idea is to learn an action-value function from which a greedy policy would yield the highest possible sum of discounted rewards. To learn such a function, this method uses the optimal Bellman operator. This operator is a contracting mapping, meaning that the successive iterations of this operator lead to its fixed point. The theory guarantees that this fixed point is the optimal action-value function corresponding to the optimal policy i.e., the policy yielding maximum reward.

DQN is known to work well on discrete action spaces. Since the action space of Pendubot is only in 1 dimension, we consider discretizing it into 9 actions. Choosing a logarithmic discretization centered on zero yields better performances in practice. This behavior can be understood because a linear discretization does not leave enough variety to balance the pendulum precisely when it is close to being upright. The proposed state space is composed of 4 dimensions. For the final submission, we used the environment settings of a competing team of Chi Zhang and Akhil Sathuluri, as their reward function yields better performances. For further details, we refer to their submission. Similar to their approach, we employ the LQR controller provided by RealAIGym to stabilize the Pendubot when it enters the region of attraction.

Algorithm 1 presents the pseudo-code of DQN. The hyperparameters can be found in Table 1.

Algorithm 1 DQN

Inputs: number of epochs N, training steps per epoch n, online and target parameters $\theta = \bar{\theta}$, replay buffer $\mathcal{D}$, gradient step frequency G, target update frequency T.
$i \leftarrow 0$ ▷ number of overall training steps
for N epochs **do**
  $j \leftarrow 0$ ▷ number of training steps within an epoch
  $s \leftarrow$ env.init()
  absorbing $\leftarrow$ false; sum_reward $\leftarrow 0$; n_episodes $\leftarrow 0$
  while $j < n$ and absorbing $=$ false **do**
    sample $a \sim \epsilon$-greedy $Q(s, \cdot|\theta)$
    $(s', r,$ absorbing$) \leftarrow$ env.step(a)
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a, r, s')\}$
    $s \leftarrow s'$; sum_reward $+= r$
    if absorbing $=$ true **then**
      $s \leftarrow$ env.init()
      n_episodes $+= 1$
    end if
    if $i = 0[G]$ **then**
      $d \sim \mathcal{U}(\mathcal{D})$
      $\theta \leftarrow$ Adam_update$(\mathcal{L}, d, \theta, \bar{\theta})$ ▷ $\mathcal{L}$=TD-error
    end if
    if $i = 0[T]$ **then**
      $\bar{\theta} \leftarrow \theta$
    end if
    $i += 1; j += 1$
  end while
end for
return θ

3 Results

Figure 1 shows how DQN manages to balance the Pendubot upright. Even if Pendubot reaches the goal under 3 seconds, the actions taken by DQN are varying a lot over the trajectory making it hard to transfer to the real system without the risk of damaging the robot. Table 2 shows the different scores that the challenge proposes to compute. The cost for the torque smoothness is the highest of the competition. Further investigation could be done to make the actions smoother over time. We believe this behavior is coming from the choice of the neural network architecture. The neural network takes as input a state and outputs the action-value function estimate for every possible action. This way the neural network does

Table 1: Summary of the hyperparameters.

γ	0.85
H	1000
# epochs N	200
# training steps per epochs n	25000
# initial samples in $\mathcal{D}$	2000
replay buffer capacity	25000
gradient step frequency G	1
target update frequency T	100
starting ϵ	1
ending ϵ	0.01
ϵ linear decay duration	25000
batch size	64
learning rate	0.00001
architecture	FC256, 256, 9

Table 2: RealAI scores.

Swingup Success	1/1
Swingup Time [s]	1.95
Energy [J]	14.44
Max. Torque [Nm]	5.0
Integrated Torque [Nms]	5.92
Torque Cost [N²m²]	25.44
Torque Smoothness [Nm]	1.452
Velocity Cost [m²/s²]	34.79
RealAI Score	0.815

not have a notion of distance between the actions. In contrast, policy gradient methods are training a neural network that directly encodes the policy. By outputting the action to apply, the policy network has a notion of distance in the action space. DQN manages to find a policy having the lowest velocity cost of the competition (see Table 2 and the leaderboard of the competition). Figure 2 shows the performances of DQN on the robustness scores. The results can be reproduced in 5 hours on a regular CPU.

4 Analysis

We choose γ to be low ($\gamma = 0.85$) compared to usual approaches. By reducing the discount factor, the agent focuses more on immediate reward, making the agent more willing to stay upright instead of waiting to reach the same state by spinning. In Figure 3, we choose two different values for γ: 0.95 and 0.99. In Figure 3b where $\gamma = 0.99$, the agent makes Pendubot spin as the values of q_1 and q_2 show. In Figure 3a where $\gamma = 0.95$, the first link of Pendubot remains upright and the agent tries to bring the second link upright as well.

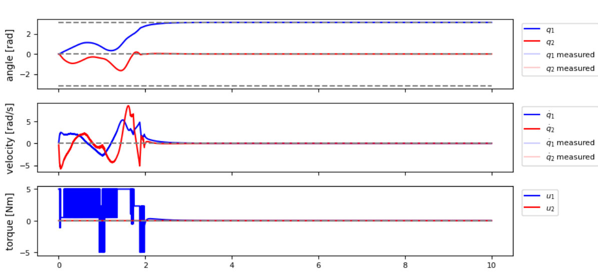


Figure 1: Position, velocity, and torque with respect to time.

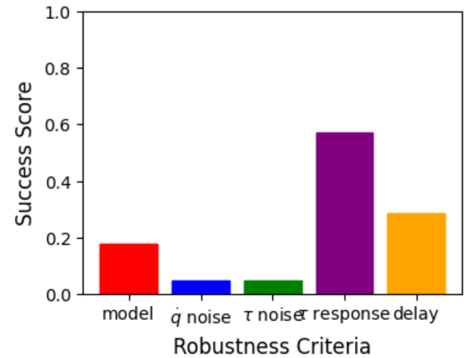
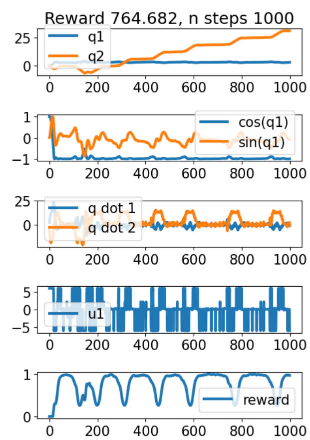
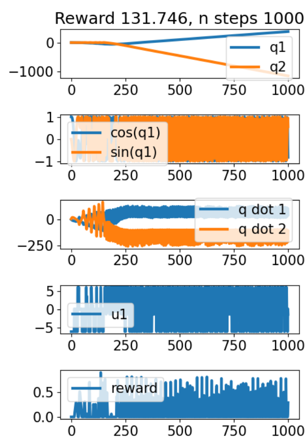


Figure 2: Robustness score. Overall Robustness Score: 0.226.



(a) $\gamma = 0.95$.



(b) $\gamma = 0.99$.

Figure 3: Position, velocity, and torque with respect to time with different γ. The X axis is in timestep. One timestep corresponds to 0.01 seconds, meaning 1000 timestep amounts to 10 seconds.

5 Conclusion

For this challenge, we tried to use DQN to solve the Pendubot task. At first sight, reinforcement learning seems to be able to solve this task, but it might not be the best approach since it requires a lot of reward and hyperparameter tuning. A more reasonable direction would be to use residual learning to learn how to compensate for disturbances with Reinforcement Learning while using control theory as guidance.

Acknowledgments

We acknowledge the grant “Einrichtung eines Labors des Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) an der Technischen Universität Darmstadt” of the Hessisches Ministerium für Wissenschaft und Kunst. Parts of the calculations for this research were conducted on the Lichtenberg high-performance computer of the TU Darmstadt. We also thank Daniel Palenicek and Tim Schneider for their support while performing our experiments on the IAS group's computing cluster at TU Darmstadt.

References

[1] N. Funk, G. Chalvatzaki, B. Belousov, and J. Peters, "Learn2assemble with structured representations and search for robotic architectural construction," in *Conference on Robot Learning*, pp. 1401–1411, PMLR, 2022.

[2] J. Degrave, F. Felici, J. Buchli, *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, no. 7897, pp. 414–419, 2022.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] F. Wiebe, S. Vyas, L. J. Maywald, S. Kumar, and F. Kirchner, "RealAIGym: Education and Research Platform for Studying Athletic Intelligence," in *Proceedings of Robotics Science and Systems Workshop Mind the Gap: Opportunities and Challenges in the Transition Between Research and Industry*, (New York), July 2022.

[5] F. Wiebe, S. Kumar, L. Shala, S. Vyas, M. Javadi, and F. Kirchner, "An open source dual purpose acrobot and pendubot platform for benchmarking control algorithms for underactuated robotics," *IEEE Robotics and Automation Magazine*, 2023. under review.