Making Deep Q-learning Methods Robust to Time Discretization

Julie Alhosh

Abstract

This paper aims to recreate the results of the paper (1) which introduces an off-policy deep reinforcement learning (DRL) algorithm that is robust to time discretization called Deep Advantage Updating (DAU). Specifically, the paper empirically proves that Q-learning methods are not robust to time discretization and highlights the need for an algorithm that can be used in tasks for near continuous environments. Thus, the paper (1) introduces DAU that achieves good performance regardless of the time discretization when tested across several gym environments. To reproduce those results, we evaluate the performance of the discrete DAU algorithm compared to the performance of the DQN algorithm (2) when solving the OpenAI's Gym Cartpole environment. Our results agree with the ones in the paper as DAU maintains a good performance when changing time discretizations compared to DQN which performs worse with smaller time discretizations.

1 Introduction

DRL algorithms have achieved remarkable results recently in a variety of domains including games such as Go (3), chess, and shogi (Japanese chess) (4), and dexterous in-hand manipulation tasks (5). However, DRL suffers from high sensitivity to several factors including changes in environment parameters and hyperparameter tuning. Due to such sensitivities, the applications of DRL are limited. For example, the ability to transfer learning from imperfect simulators to real-world settings is crucial in robotics but DRL cannot be used to achieve this goal as it is sensitive to small changes in environment parameters. Moreover, in near-continuous environments, Q-learning-based approaches tend to be sensitive to time discretization. As near-continuous time environments include most continuous control environments and robotics, it is of interest to have a DRL algorithm that is robust to time discretization. Although, there are a few on-policy DRL algorithms that can be setup to be robust against time discretization including PPO (6), TRPO (7) and A3C (8), the paper (1) focuses on off-policy algorithms such as DQN (2) and DDPG (9). The paper (1) introduces DAU, an off policy algorithm, and provides empirical evidence for its robustness to time discretization. In this paper, we show that our results for the discrete action space environment, Cartpole, are consistent with the results of the original paper providing further support to the deep advantage updating schema introduced in (1) since the findings generalize to a different implementation.

2 Background

2.1 Continuous-time and near-continuous Markov Decision Processes

Let $S = \mathbb{R}^d$ be a set of states, and $\mathcal{A}$ be a set of actions. A *continuous-time Markov Decision Process (MDP)* is defined by the differential equation

$$\frac{ds_t}{dt} = F(s_t, a_t) \tag{1}$$

where $F : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a function that describes the dynamics of the environment.

For any timestep $\delta t > 0$, a discretization of the continuous-time MDP with time discretization δt is defined by

$$\mathcal{M}_{\delta t} = \langle \mathcal{S}, \mathcal{A}, T_{\delta t}, r_{\delta t}, \gamma_{\delta t} \rangle$$

where $T_{\delta t}$ is the transition function. The transition function of a state s is the state obtained when starting at $s_0 = s$ and maintaining the action $a_t = a$ constant for a time period δt which corresponds to an agent evolving in the continuous environment as the one defined by (1), but only making observations and choosing actions every δt (1). The discount factor and reward are defined as follows

$$\gamma_{\delta t} := \gamma^{\delta t}$$
$$r_{\delta t} := \delta t . r$$

where γ, r are the discount factor and return of the continuous-time MDP.

We call the MDP defined above $\mathcal{M}_{\delta t}$ *near-continuous MDP*.

2.2 Q-learning in continuous and near-continuous time

The papers (10; 1) provide formal proof that the Q-function collapses to the value function in continuous time (See Theorem 1), thus making it impossible to infer actions based on the Q-function. Although the Q-function would still depend on actions in near-continuous time when the time discretization, δt is strictly positive, the approximation error of the Q-function would be larger than the effect of individual actions on the Q-function. This is the theoretical reasoning as to why the Q-learning-based methods are sensitive to time discretizations and do not perform as well when δt approaches 0.

Theorem 1 *(From (1)) Under suitable smoothness assumptions, The action-value function of a near-continuous MDP is related to its value function via*

$$Q^{\pi}_{\delta t}(s, a) = V^{\pi}_{\delta t}(s) + O(\delta t) \tag{2}$$

when $\delta t \to 0$, for every $(s, a) \in (\mathcal{S} \times \mathcal{A})$.

2.3 Advantage updating and DAU

A rescaled version of the advantage function (3) provides the same information on actions as the Q-function and does not collapse to the value function in the continuous-time limit.

$$A^{\pi}_{\delta t}(s, a) := \frac{Q^{\pi}_{\delta t}(s, a) - V^{\pi}_{\delta t}(s)}{\delta t} \tag{3}$$

Under suitable assumptions, the rescaled version of the advantage function allows us to write the discretized version of the Q-function as

$$Q^{\pi}_{\delta t}(s, a) = V^{\pi}_{\delta t}(s) + \delta t A^{\pi}_{\delta t}(s, a) \tag{4}$$

The DAU algorithm introduced in (1), trains two networks, V_θ which approximates the value function $V^{\pi}_{\delta t}$, and A_ψ to approximate the rescaled advantage function $A^{\pi}_{\delta t}$. The algorithm uses those networks to compute the action-value function during training. Furthermore, the following equation needs to be satisfied to guarantee the stability of A_ψ when $\delta t \to 0$

$$A_\psi(s, \pi(s)) = 0 \tag{5}$$

In the original paper (1), they introduce a parametric function $\bar{A}_\psi$ to define A_ψ as

$$A_\psi(s, a) := \bar{A}_\psi(s, a) - \bar{A}_\psi(s, \pi(s)) \tag{6}$$

Since the continuous time limit of the rescaled advantage function depends on actions, this approach overcomes the issue encountered in standard Q-learning-based approaches.

3 Methodology

We test the performance of DQN (2), an off-policy algorithm for environments with a discrete action space and the discrete version of DAU (1) on a modified Cartpole environment using two values for time discretizations $\delta t \in \{0.01, 0.001\}$. It is important to state that as δt decreases, the run time of an experiment for the same physical time increases by a factor of $\frac{1}{\delta t}$ which is why we only ran experiments on two δt values. Also, in our implementation, we used smaller networks compared to the ones used in the original paper to favor smaller run time and we were able to do so because Cartpole is a relatively easy environment to learn.

Considering that the default value for the timestep between state updates in the Cartpole environment is 0.02[1], the δt values we chose to test the algorithms DQN and DAU cover a range of values that are sufficient to check whether the algorithms are robust to time discretization. We modify the OpenAI's Gym Cartpole environment to accommodate for different time discretizations which includes changing the timestep between state updates, the reward threshold, and the maximum number of steps.

To implement DQN, we start by following a tutorial[2] and the corresponding implementation[3]. We continue to modify the implementation mainly to allow for different time discretizations which include scaling the hyperparameters and the reward. The pseudo-code of the resulting DQN implementation is shown in Algorithm 1. As for the discrete DAU algorithm, we implement it following the pseudo-code shown in Algorithm 2.

Algorithm 1 DQN for time discretization δt

```
Input:
$\gamma$ discount factor.
$\alpha$ learning rate.
$\delta t$ time discretization.
n_update target network update frequency.

1: Initialize $\psi$ parameter of network $Q_\psi$.
2: Copy $\psi$ parameters to $\psi'$ parameter of target network $Q'_\psi$.
3: Initialize $\mathcal{D}$, a replay buffer.
4: opt an optimizer for the network $Q_\psi$ with learning rate $\delta t * \alpha$.
5: for $i = 1, 2, \ldots, N_{train}$ do
6:     if $i$ mod n_update $= 0$ then
7:         $\psi' \leftarrow \psi$
8:     end if
9:     Observe initial state $s = s^0$
10:    while $s'$ is not None, do
11:        $a = \pi^{\epsilon-greedy}(s)$
12:        Perform $a$ and observe $(r', d', s')$.
13:        Store $(s, a, r', d', s')$ in $\mathcal{D}$.
14:        $s \leftarrow s'$
15:        If the size of $\mathcal{D}$ is sufficient, sample a batch of $N$ random transitions from $\mathcal{D}$.
16:        $\tilde{Q}^i \leftarrow r^i \delta t + (1 - d^i)\gamma^{\delta t} \max_{a'} Q_{\psi'}(s^{'i}, a')$
17:        $\Delta\psi \leftarrow \frac{1}{N}\sum_{i=0}^{N} \left(Q^i - \tilde{Q}^i\right)\partial_\psi Q_\psi(s^i, a^i)$
18:        Update $\psi$ with opt, $\Delta\psi$ and learning rate $\delta t * \alpha$.
19:    end while
20: end for
```

The main difference between the two implementations is that DQN keeps track of and updates a Q-network and a target Q-network, where the weights of the target Q-network are updated every **n_update** training iteration whereas the DAU algorithm keeps track of and updates a value function network, V_θ, and an advantage function network, A_ψ. All networks used in both algorithms share the same structure and use RMSProp as an optimizer. Moreover, both algorithms scale the reward obtained, the discount factor, and the learning rate according to the time discretization δt. For each time discretization

[1]Information available at: https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py
[2]Available at: https://towardsdatascience.com/deep-q-learning-for-the-cartpole-44d761085c2f
[3]Available at: https://github.com/ritakurban/Practical-Data-Science/blob/master/DQL_CartPole.ipynb

$\delta t \in \{0.01, 0.001\}$ and for each algorithm, the learning curve is averaged over two independent runs. The learning curves for DQN and DAU are presented in Figures 1 and 2 respectively.

Algorithm 2 Discrete DAU for time discretization δt

```
Input:
$\gamma$ discount factor.
$\delta t$ time discretization.
$\alpha$ learning rate.

 1: Initialize $\theta, \psi$ parameters of the networks $V_\theta, \bar{A}_\psi$.
 2: Initialize $\mathcal{D}$, a replay buffer.
 3: **opt**$_V$, **opt**$_{\bar{A}}$ optimizers for the networks $V_\theta, \bar{A}_\psi$ respectively with learning rate $\delta t * \alpha$.
 4: for $i = 1, 2, \dots, N_{train}$ do
 5:     Observe initial state $s = s^0$
 6:     while $s'$ is not None, do
 7:         $a = \pi^{\epsilon-greedy}(s)$
 8:         Perform $a$ and observe $(r', d', s')$.
 9:         Store $(s, a, r', d', s')$ in $\mathcal{D}$.
10:         $s \leftarrow s'$
11:         If the size of $\mathcal{D}$ is sufficient, sample a batch of $N$ random transitions from $\mathcal{D}$.
12:         $Q^i \leftarrow V_\theta(s^i) + \delta t\big(\bar{A}_\psi(s^i, a^i) - \max_{a'} \bar{A}_\psi(s^i, a')\big)$
13:         $\tilde{Q}^i \leftarrow r^i \delta t + (1 - d^i)\gamma^{\delta t} V_\theta(s'^i)$
14:         $\Delta\theta \leftarrow \frac{1}{N}\sum_{i=0}^{N} \frac{(Q^i - \tilde{Q}^i)\partial_\theta V_\theta(s'^i)}{\delta t}$
15:         $\Delta\psi \leftarrow \frac{1}{N}\sum_{i=0}^{N} \frac{(Q^i - \tilde{Q}^i)\partial_\psi\left(\bar{A}_\psi(s^i, a^i) - \max_{a'} \bar{A}_\psi(s^i, a')\right)}{\delta t}$
16:         Update $\theta$ with **opt**$_V$, $\Delta\theta$ and learning rate $\delta t * \alpha$.
17:         Update $\psi$ with **opt**$_{\bar{A}}$, $\Delta\psi$ and learning rate $\delta t * \alpha$.
18:     end while
19: end for
```

4 Experimental Results

As discussed in the introduction, it is vital for many near-continuous environments including continuous control environments and robotics that the algorithms used to solve the environment are robust to time discretization. We test the proposed algorithm, discrete DAU, against the DQN algorithm by running two independent experiments for each value $\delta t \in \{0.01, 0.001\}$ to train both algorithms on the Cartpole environment for 180 episodes each. We plot the average returns of those experiments for each algorithm along with the standard deviation against the physical time in hours in Figures 1 and 2.

For the DQN algorithm (see Figure 1), we notice that the algorithm learns well when $\delta t = 0.01$ as it achieves returns of approximately 200 after learning for 0.05h in physical time. However, its performance decreases significantly as the time discretization drops to $\delta t = 0.001$ where it achieves returns of approximately a 100. As for the discrete DAU algorithm (see Figure 2), we can see that algorithm learns well similar to the performance of DQN when $\delta t = 0.01$ as it achieves returns of approximately 200 after learning for 0.04h in physical time and achieves a slightly better performance as the time discretization drops to $\delta t = 0.001$ where it achieves returns of approximately 225 after learning for about 0.09h in physical time.

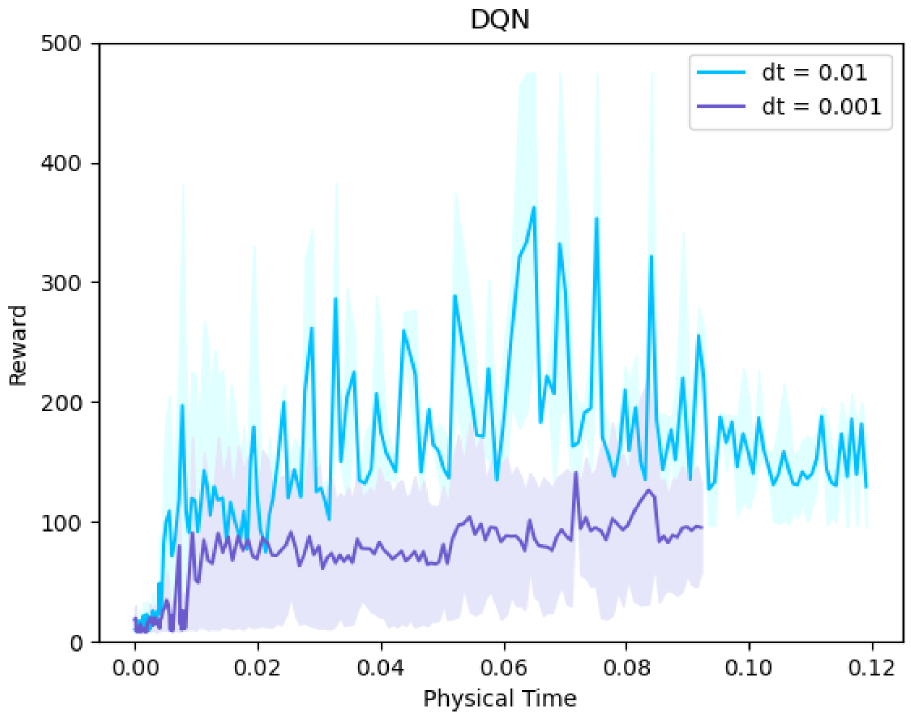

Figure 1: Learning curve for DQN on the Cartpole environment.

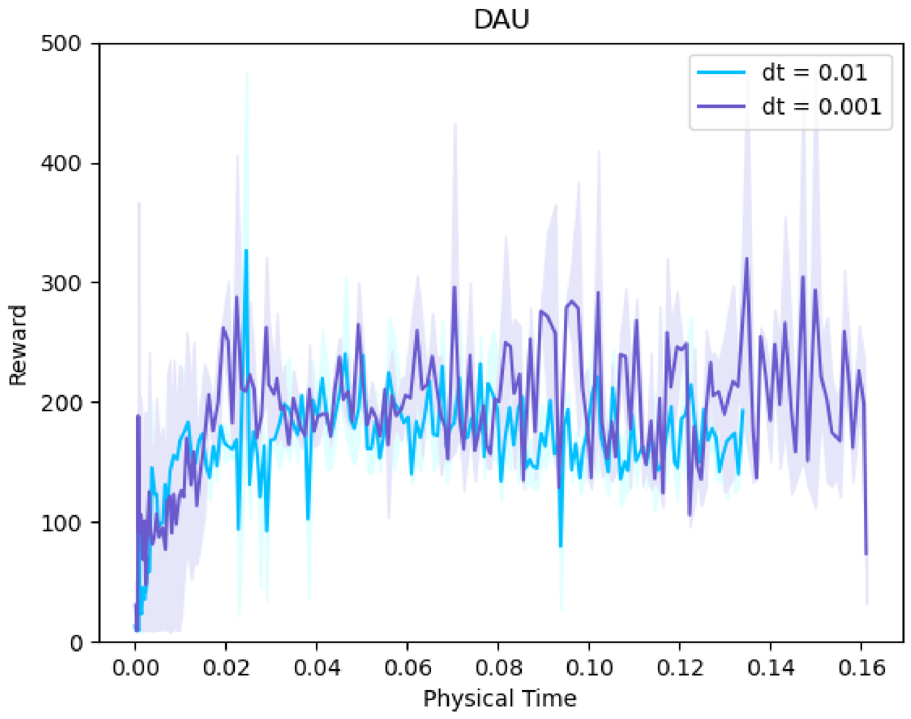

Figure 2: Learning curve for discrete DAU on the Cartpole environment.

These results show that the discrete DAU algorithm learns good policies regardless of the time discretization whereas the DQN algorithm fails to adapt to a smaller time discretization. Although we used different implementations than the ones used in the original paper, our results agree with the results of the paper (1) adding further support to their results as discrete DAU is shown to be robust to time discretization and implementation details.

5 Conclusion

The purpose of this paper was to reproduce the findings found in (1), which showed that off-policy Q-learning-based DRL algorithms are not robust against time discretization and introduced an off policy algorithm that is robust to time discretization. Thus, we conducted experiments on the Cartpole environment which included training the DQN and discrete DAU algorithms using two time discretizations. We found that the DQN algorithm fails to learn a good policy as the time discretization decreases while the discrete DAU algorithm is much more robust to small time discretization.

References

[1] C. Tallec, L. Blier, and Y. Ollivier, “Making deep q-learning methods robust to time discretization,” *CoRR*, vol. abs/1901.09732, 2019.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, 2015.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. v. d. Driessche, T. Graepel, and D. Hassabis, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, “Mastering chess and shogi by self-play with a general reinforcement learning algorithm,” *CoRR*, vol. abs/1712.01815, 2017.

[5] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, “Learning dexterous in-hand manipulation,” *CoRR*, vol. abs/1808.00177, 2018.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.

[7] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” *CoRR*, vol. abs/1502.05477, 2015.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 1928–1937, PMLR, 20–22 Jun 2016.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[10] L. C. Baird, “Reinforcement learning in continuous time: advantage updating,” *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN’94)*, vol. 4, pp. 2448–2453 vol.4, 1994.