# Reinforcement Learning Report for TD3 applied to the Reacher-v2 environment

Olga Sorokoletova Giacomo Telloli
1937430 - 1792451
Sapienza Universitá di Roma

February 19, 2021

**Abstract**

*Twin delayed deep deterministic policy gradient algorithm (TD3) proposes a deterministic algorithm that substantially improves on DDPG. We have considered the main weaknesses of the latter which were eliminated by newly introduced components of the former. The report contains the description of these components, analysis of the contribution of each of them to the final improvement of TD3 over DDPG and some qualitative plots employed for the representation of the improvements and obtained by the suite of our experiments. An evaluation procedure was performed on the OpenAI gym task Reacher-V2.*

## I. Project Description

The underlined idea of our project is based on the published by the authors of TD3 paper [1] (Fujimoto, S., van Hoof, H., and Meger, D.), and also on the papers devoted to the general concepts of the Deep Deterministic Policy Gradient (DDPG) and (Soft) Actor-Critic (SAC) ([2] and [3]) as well as using self-tuned implementation of the open source code provided in the [1]: https://github.com/sfujim/TD3 to explore the new possibilities given by the TD3 method in resolving issues incidental to DDPG and inherently improving its performance and speed of the learning. In parallel we are exploring and discussing cited issues by themselves and trying to recreate some experiments proposed by the authors applying them to Reacher-V2 and prove an effectiveness. In particular, Twin Delayed algorithm focuses on two outcomes that occur as the result of estimation error, namely overestimation bias and a high variance build-up, and the next section of our report takes them into precise consideration.

## II. Environment

OpenAI Gym includes an environment Reacher-V2 - a robot arm in a 2D space which goal is to reach a target.

The environment is episodic with a continuous action space and a continuous state space.

The state space is made by a 11-dimensional vector:

$$o = \begin{bmatrix} \cos\theta_1 \\ \cos\theta_2 \\ \sin\theta_1 \\ \sin\theta_2 \\ \theta_1 \\ \theta_2 \\ \theta_1' \\ \theta_2' \\ \text{diff} \\ \text{done} \end{bmatrix} \tag{1}$$

where $diff$ is a 2-dimensional vector - the distance between the fingertip and target point, and *done* is a Boolean variable which takes *True* if the fingertip has reached the target.

The action to be performed is to *add* or to *subtract* at maximum 1 radiant at each joint.

The reward is calculated with the formula:

$$Reward = -(rewardDist + rewardCtrl) \tag{2}$$

where *rewardDist* is a $diff$ normalized, and *rewardCtrl* is the squared sum of the action parameters.

The initial angles and velocities of the joints are $[0, 0]$ and $[0, 0]$, and the goal is a random position in the plane (the position of a target).

The graphical representation of the environment sample is illustrated in the Figure 1.

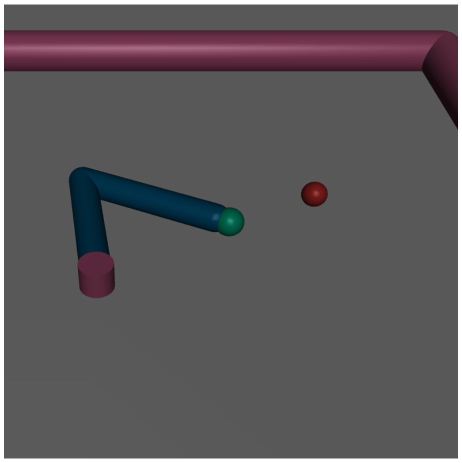
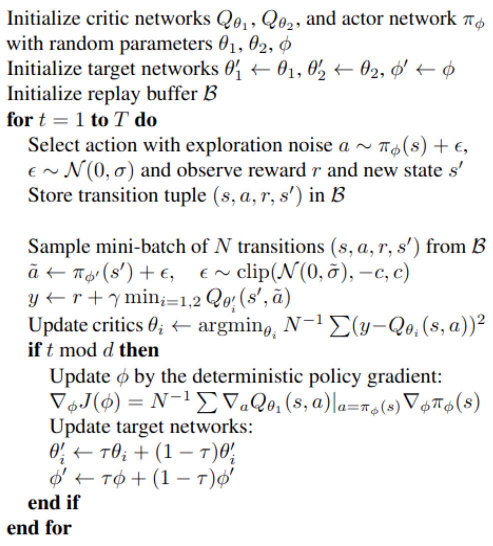


**Figure 1:** *OpenAI Gym Reacher-v2*

## III. Algorithm

Twin Delayed Deep Deterministic Policy Gradient is a variation of DDPG algorithm that includes three crucial modifications: Clipped Double Q-learning, delayed policy updates and target policy smoothing. So, we are facing actor-critic setting that includes listed components (as follows, we have two estimators for the two Q-functions $Q_{\theta_1}$ and $Q_{\theta_2}$ and one policy estimator $\pi_\phi$). The pseudo-code used as a baseline for the implementation of the highlighted ideas can be found below.



```
Initialize critic networks $Q_{\theta_1}$, $Q_{\theta_2}$, and actor network $\pi_\phi$
with random parameters $\theta_1$, $\theta_2$, $\phi$
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ to $T$ do
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$,
  $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward $r$ and new state $s'$
  Store transition tuple $(s, a, r, s')$ in $\mathcal{B}$

  Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if $t$ mod $d$ then
    Update $\phi$ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for
```

**Figure 2:** *TD3 algorithm*

### i. Overestimation Bias and Clipped Double Q-learning

The first problem TD3 algorithm is intended to deal with by the clipped variant of Double Q-learning is an overestimation bias in an actor-critic setting (since it has been proven in the corresponding researches that value estimate in deterministic policy gradients will be an overestimation).

The policy is to be updated with respect to the value estimates of an approximate critic. And although this overestimation may be minimal with each particular update, an overestimation may accumulate and eventually develop into a significant bias over many updates. Consequently, an inaccurate value estimate may lead to poor policy updates, i.e. suboptimal actions might be highly rated by the suboptimal critic implying the suboptimal action choice in the next policy update - so-called feedback loop.

In contrast to DDPG, which is well-known to suffer from overestimation bias problem, it can be clearly seen from Figure 3 that TD3 is able to significantly reduce this error.

We are plotting the value estimate of TD3 over time while it learns on the OpenAI gym environment Reacher-V2. The red curve corresponds to the average value estimate over 10000 states, the blue one - to the true value. True value is estimated using the average discounted return ($\gamma = 0.99$) over 1000 episodes following the current policy, starting from states sampled from the replay buffer. The total number of time steps is set to be equal $5 * 10^5$.

The idea behind the algorithm for overestimation bias handling is a Clipped Double Q-learning - clipped variant of Double Q-learning, which can replace the critic in any actor-critic method.

The original Double Q-learning uses a pair of actors $(\pi_{\phi_1}, \pi_{\phi_2})$ and pair of critics $(Q_{\theta_1}, Q_{\theta_2})$, where $\pi_{\phi_1}$ is optimized with respect to $Q_{\theta_1}$, and $\pi_{\phi_2}$ with respect $Q_{\theta_2}$:

$$y_1 = r + \gamma Q_{\theta'_2}(s', \pi_{\phi_1}(s')) \qquad (3)$$

$$y_2 = r + \gamma Q_{\theta'_1}(s', \pi_{\phi_2}(s')). \qquad (4)$$

In the Clipped Double Q-learning instead upper-bounding the less biased value estimate $Q_{\theta_2}$ by the biased estimate $Q_{\theta_1}$ is proposed.

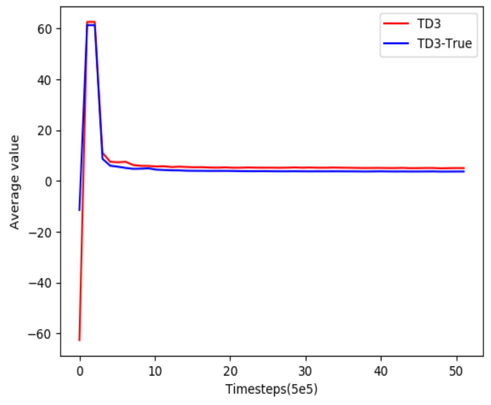


**Figure 3:** *Measuring overestimation bias in the value estimates of TD3*

This results in taking the minimum between the two estimates:

$$y_1 = r + \gamma \min Q_{\theta'_i}(s', \pi_{\phi_1}(s')) \tag{5}$$

The same target $y_2 = y_1$ for $Q_{\theta_2}$ is then used.

## ii. High Variance and Delayed Policy Updates with Target Networks

High variance estimates provide a noisy gradient for the policy update. Therefore, it was important to propose modifications to the learning procedure of actor-critic for variance reduction so that error could be minimized at each update.

Due to the temporal difference update, there is a build up of error. In a function approximation setting the Bellman equation is never exactly satisfied, and each update leaves some amount of residual TD-error. Variance of the estimate can grow rapidly (given large $\gamma$), because it will be proportional to the variance of future reward and estimation error.

Fortunately, the use of a target networks reduces the growth of error. As deep function approximators require multiple gradient updates to converge, target networks provide a stable objective in the learning process.

To provide examine the learning behavior with and without target networks on both the critic and actor we draw Figure 4 and Figure 5 in the Reacher-V2 environment. In the former we compare the behavior with a fixed policy and in the latter - with a policy that continues to learn, trained with the current value estimate. The goal of plotting both situations is to observe the influence of the error when paired with a policy maximizing over the value estimate.

The target networks use an update rate, parametrized by $\tau$. Value estimate is again averaged over 10000 states. The total numbers of time steps are set to be equal $5 \times 10^5$ and $2 \times 10^5$ correspondingly.

In the regular case, all update rates result in similar convergent behaviors when considering a fixed policy, and the use of fast-updating ($\tau = 1$) target networks results in a divergent behavior, when the policy is trained. However, results obtained by the experiments in the Reacher-V2 environment seem to be not so representative in that sense.

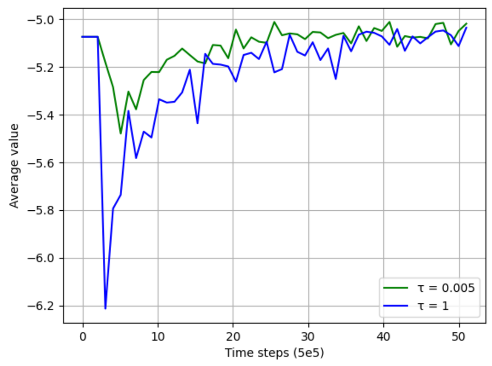


**Figure 4:** *Average estimated value without target networks, ($\tau = 1$), and with slow-updating target networks, ($\tau = 0.005$), with a fixed policy*

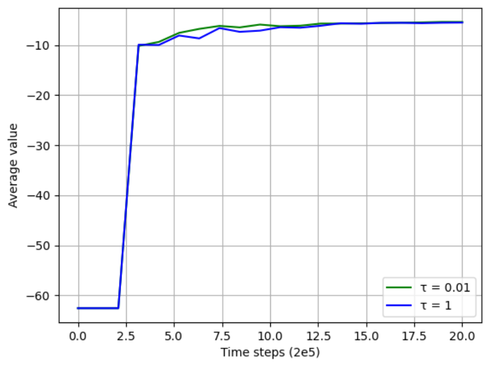


**Figure 5:** *Average estimated value without target networks, ($\tau = 1$), and with slow-updating target networks, ($\tau = 0.01$), with a learned policy*

If target networks can be used to reduce the error over multiple updates, and policy updates on high-error states cause divergent behavior, then the policy network should be updated at a lower frequency than the value network, to first minimize error before introducing a policy update - delaying policy updates until the value error is as small as possible. The modification is to only update the policy and target networks after a fixed number of updates $d$ to the critic.

The hyper-parameter $d$ should be chosen appropriately, since training the actor for too few iterations would cripple learning. In the Figure 6 different choices for the Policy Frequency parameter $d$ are performed. Assuming $\tau = 0.005$, the value of $d = 2$ is found to be the best.

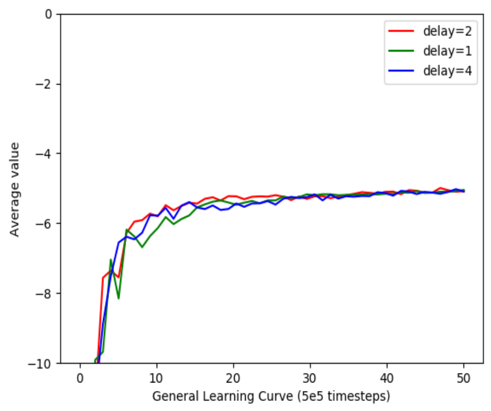


**Figure 6:** *Average return of* $5 \times 10^5$ *time steps for settings of d to be equal respectively: red curve - 2, blue curve - 4, green curve - 1 (without delaying)*

### iii. Target Policy Smoothing Regularization

When updating the critic, a learning target using a deterministic policy is highly susceptible to inaccuracies induced by function approximation error, increasing the variance of the target. This induced variance can be reduced through regularization. Target Policy Smoothing (TPS) Regularization technique was introduced into TD3 to address this problem. The underlined idea is that fitting the value of a small area around the target action would have the benefit of smoothing the value estimate by bootstrapping off of similar state-action value estimates. This modifies the target update in such a way so the additional noise is clipped to keep the target close to the original action:

$$\epsilon \sim clip(\mathcal{N}(0, \sigma), -c, c) \tag{6}$$

Some suggestions about the contribution of a such smoothing regularization strategy into final performance of an entire algorithm can be found in the subsection Results of the next section (TD3-TPS).

## IV. Evaluation

Our evaluation procedure is primarily built on the idea of the comparison between the learning curves of the DDPG algorithm taken as a baseline, complete version of the TD3 algorithm and versions of TD3 without each of the main three ingredients of it: Clipped Double Q-learning, delayed policy updates and target policy smoothing in the environment Reacher-V2. We present our results with the full graph and summarized table in the corresponding subsection below where we compare the performance of removing each component from TD3 along with modifications proposed to the architecture.

### i. Hyper-parameters Setting

Following to [1] a set of hyper-parameters proposed as being optimal («standard») and listed in the tables Table 1 and Table 2 has been used during runs of experiments everywhere except for the cases in which the tuning of a particular hyper-parameter is a sense of the experiment by itself.

| **Hyper-parameter** | **Value** |
|---|---|
| Critic Learning Rate | $3 \times 10^{-4}$ |
| Critic Regularization | None |
| Actor Learning Rate | $3 \times 10^{-4}$ |
| Actor Regularization | None |
| Optimizer | Adam |
| Target Update Rate ($\tau$) | $5 \times 10^{-3}$ |
| Batch Size | 256 |
| Policy Noise | $\mathcal{N}(0, 0.2)$ |
| Discount Factor | 0.99 |
| Policy Frequency (d) | 2 |
| Normalized Observations | False |
| Noise Clip | 0.5 |
| Exploration Policy | $\mathcal{N}(0, 0.1)$ |

**Table 1:** *«Standard» hyper-parameter choices.*

| Critics | Actor |
|---|---|
| (S + A, 256) | (S, 256) |
| ReLU | ReLU |
| (256, 256) | (256, 256) |
| ReLU | ReLU |
| (256, 1) | (256, A) |
| Identity | Tanh |

**Table 2:** *Neural Networks Architecture (both Critics have the same one). S - state dim, A - action dim.*

### ii. Results

The general form of the learning curve for the OpenAI gym continuous control task Reacher-V2 while applying complete TD3 algorithm with the hyper-parameters listed above is represented in the Figure 7. Here we perform average evaluation over 10 trials $5 \times 10^5$ (half of a million) time steps. This choice of an overall number of steps is motivated by the reduction of the number of time steps used by authors of the method for the evaluation procedure by a factor of 2 due to the restricted capabilities of our hardware. But still we can observe that curvature obtained by our implementation is almost coincident with the curvature of the plot obtained in [1] for the Reacher-V1 over 1 million time steps which is a relative task.

Max Average Return over 10 trials of 0.5 million time steps we got for Reacher-V2 in TD3 standard settings equals to -6.4.

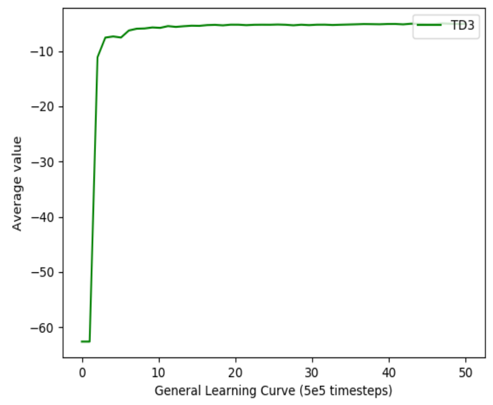


**Figure 7:** *Learning curve of TD3 for Reacher-V2*

The significance of each component of the TD3 can be estimated through the drawing of the comparative graph 8. As was expected, the full algorithm outperforms every other combination in our task (as well as on the most of tasks to which it has been applied by scientific community).

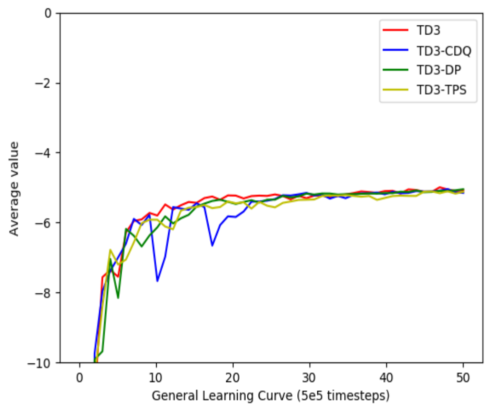


**Figure 8:** *Comparing learning curves of complete TD3 with eliminations one by one of its distinguishing features: delayed policy updates (TD3-DP), target policy smoothing (TD3-TPS), Clipped Double Q-learning (TD3-CDQ)*

Taken together, features of the Twin Delayed Deep Deterministic policy gradient approach (TD3) provide both the learning speed and performance improvements over DDPG.

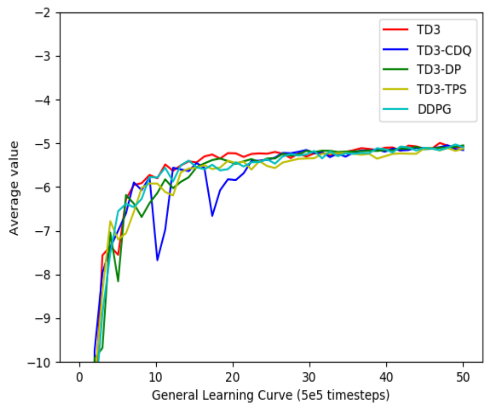


**Figure 9:** *Comparison of TD3 settings with the DDPG baseline*

Final results - average returns over 10 trials of 0.5 million time steps for all settings represented in the Figure 9 - are summarized in the table Table 3.

Although growth of the performance of the TD3 over DDPG does not seem really sig-

| Method | Reacher-V2 |
|---|---|
| TD3 | -6.4 |
| TD3-CDQ | -9.1 |
| TD3-DP | -9.0 |
| TD3-TPS | -9.1 |
| DDPG | -7.0 |

**Table 3:** *Average return. Effectiveness of TD3 approach and components of it.*

nificant according to the experiments in the Reacher-V2 environment, it can be clearly seen from the 3 that the addition of all three components instead of only a single one is crucial for reducing of negative influences that comes from Overestimation Bias and High Variance problems.

Note: the results of exceeding in performance and learning speed over DDPG also depend on the architecture and hyper-parameter decisions for the latter. In experiments presented in this report we have been mostly using settings proposed by [2].

## V. Conclusion

Based on the referenced articles, we have been implemented some experimental analysis devoted to the exploration of the effectiveness of the Tween Delayed Deep Deterministic Policy Gradient algorithm in the Reacher-V2 environment. In particular, we have measured how it addresses the Overestimation Bias and High Variance challenges that have not been handled in the standard DDPG setting and compared the algorithm with its modified versions and DDPG baseline.

Summarizing everything, the following considerations can be derived: Appropriately tuned implementation of TD3 algorithm helps to invalidate errors originated by Overestimation Bias and High Variance issues in the Reacher-V2 environment, although the latter one is slightly affected; TD3 is a self-contained algorithm and all its newly introduced components: Clipped Double Q-learning, delayed policy updates and target policy smoothing contribute to outperforming over the previous state-of-the-art approach for relative tasks - DDPG.

Conclusions presented in this section are reinforced by graphical data included into report.

## References

[1] *Fujimoto, S., van Hoof, H., and Meger, D.* Addressing function approximation error in actor-critic methods. // arXiv preprint arXiv:1802.09477, 2018.

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

[3] *Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S.* Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. // arXiv preprint arXiv:1801.01290, 2018.