OFF-POLICY MULTI-STEP Q-LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In the past few years, off-policy reinforcement learning methods have shown promising results in their application for robot control. Deep Q-learning, however, still suffers from poor data-efficiency which is limiting with regard to real-world applications. We follow the idea of multi-step TD-learning to enhance data-efficiency while remaining off-policy by proposing two novel Temporal-Difference formulations: (1) Truncated Q-functions which represent the return for the first $n$ steps of a policy rollout and (2) Shifted Q-functions, acting as the farsighted return after this truncated rollout. We prove that the combination of these short- and long-term predictions is a representation of the full return, leading to the Composite Q-learning algorithm. We show the efficacy of Composite Q-learning in the tabular case and compare our approach in the function-approximation setting with TD3, Model-based Value Expansion and TD3($\Delta$), which we introduce as an off-policy variant of TD($\Delta$). We show on three simulated robot tasks that Composite TD3 outperforms TD3 as well as state-of-the-art off-policy multi-step approaches in terms of data-efficiency.

1 INTRODUCTION

In recent years, Q-learning (Watkins and Dayan, 1992) has achieved major successes in a broad range of areas by employing deep neural networks (Mnih et al., 2015; Silver et al., 2018; Lillicrap et al., 2016), including environments of higher complexity (Riedmiller et al., 2018) and even in first real world applications (Haarnoja et al., 2019). Due to its off-policy update, Q-learning can leverage transitions collected by any policy which makes it more data-efficient compared to on-policy methods. Deep Q-learning, however, still has a very high demand for data samples which is limiting with regard to robot applications. One reason for the low data-efficiency is the long temporal horizon the reward signal has to propagate through. Data-efficiency of on-policy Temporal-Difference methods can be enhanced by the use of $n$-step returns, where a Monte Carlo rollout of length $n$ is combined with a bootstrap of the value function. To employ $n$-step returns in an off-policy setting, subtrajectories of the exploratory policy have to be stored. These stored $n$-step returns, however, will differ from the true value of the target-policy. In order to benefit from $n$-step data, the replay buffer has to be restricted in size or $n$ has to be set to a small value to keep the samples close to the target-policy (Barth-Maron et al., 2018; Hessel et al., 2018). To avoid these problems, a dynamics model can be used for imaginary rollouts, the so-called Model-based Value Expansion (MVE) (Feinberg et al., 2018). Alternatively, the full return can be composed of value functions with increasing discount, an approach called TD($\Delta$) (Romoff et al., 2019). In this work, we define a model-free Temporal-Difference formulation which follows the idea of multi-step learning while remaining off-policy.

Our contributions are threefold. First, we introduce the Composite Q-learning algorithm. For its formulation, we define Truncated Q-functions, representing the return for the first $n$ steps of a policy rollout. In addition, we introduce Shifted Q-functions which represent the farsighted return after this truncated rollout. Both are then combined in a mutual recursive definition of the Q-function for the final algorithm. Second, we evaluate MVE within TD3, leading to MVE-TD3. And third, we introduce TD3($\Delta$), an extension of TD($\Delta$) to deep Q-learning. We discuss related work in Section 2, describe the theoretical background in Section 3, and define Composite Q-learning, MVE-TD3 and TD3($\Delta$) in Section 4. By breaking down the long-term return into a composition of several short-term predictions, our method increases data-efficiency which we show in the tabular case and for three simulated robot tasks in Section 5. We then conclude in Section 6.
2 Related Work

In order to correct for the deviation from the current target-policy, several methods suggest the use of Importance Sampling (Precup et al. 2000, 2001; Munos et al. 2016). Approaches based on Importance Sampling, however, can come with a vast increase in variance or can be of high cost and are not easily applicable to deterministic policies. One way to remain off-policy in multi-step Q-learning is to get the Monte Carlo rollout on the basis of the current target-policy applied to a learned dynamics model (Feinberg et al., 2018; Buckman et al., 2019). Due to accumulating errors of single-step models, this can lead to severe stability issues in the Q-update, which is also the conclusion of Feinberg et al. (2018). The authors suggest to average all intermediate i-step returns to smoothen out the model error, the so-called TD-k trick. In contrast to our work, Feinberg et al. assume to have access to the true reward function. In order to balance the length of the model-assisted rollout, Buckman et al. (2018) couple it with an uncertainty estimate from value function and model ensembles. We alleviate the problem of accumulating error by estimating a multi-step dynamics model implicitly via consecutive bootstrapping. Most related to our approach is TD(Δ) (Romoff et al., 2019). Romoff et al. formalize a Bellman-operator over the differences between value functions of increasing discount values. Their approach is on-policy and can therefore benefit directly from n-step returns. We extend TD(Δ) to the off-policy case below. Q-learning can also be extended to a SARSA-like tree-backup called $Q(\sigma)$ (Asis et al., 2018; Hernandez-Garcia and Sutton, 2018). However, it is an open question how to adjust this idea to continuous action-spaces and deterministic policies. In the Hybrid Reward Architecture (HRA), van Seijen et al. (2017) suggest a decomposition of the reward and the estimation of value functions for each part of this decomposition which are then combined as an approximation of the full return. HRA addresses the problem of complex rewards and is thus complementary to our work focusing on long time scales.

3 Background

We consider tasks modelled as Markov decision processes (MDP), where an agent executes action $a_t \in A$ in some state $s_t \in S$ following its stochastic policy $\pi$. According to the dynamics model $M$ of the environment, the agent transitions into some state $s_{t+1} \in S$ and receives scalar reward $r_t$. The agent aims at maximizing the expected long-term return $R^\pi(s_t) = \mathbb{E}_{a_{j+1} \sim \pi, s_{j+1} \sim M[\sum_{j=t}^{T-1} \gamma^j r_{j+1}]}$, where $T$ is the (possibly infinite) temporal horizon of the MDP and $\gamma \in [0,1]$ the discount factor. It therefore tries to find $\pi^\star$, s.t. $R^{\pi^\star} \geq R^\pi$ for all $\pi$. If the model of the environment is unknown, model-free methods based on the Bellman Optimality Equation over the so-called action-value $Q^\pi(s_t,a_t) = \mathbb{E}_{s_{t+1} \sim M[\sum_{j=t}^{T-1} \gamma^j r_{j+1}]}|s_t,a_t]$ can be used. In the following, we abbreviate $\mathbb{E}_{a_{j+1} \sim \pi, s_{j+1} \sim M[\sum_{j=t}^{T-1} \gamma^j r_{j+1}]}|s_t,a_t]$ by $\mathbb{E}_{s \sim M[\cdot]}$. One popular representative of continuous model-free reinforcement learning is the Deep Deterministic Policy Gradient algorithm (DDPG) (Lillicrap et al., 2016). In DDPG, actor $\mu$ is a deterministic mapping from states to actions, $\mu : S \mapsto \mathbb{A}$, representing the actions that maximize the critic $Q^\mu$, i.e. $\mu(s_t) = \arg \max_a Q^\mu(s_t,a)$. $Q$ and $\mu$ are estimated by function approximators $Q(\cdot; \theta^Q)$ and $\mu(\cdot; \theta^\mu)$, parameterized by $\theta^Q$ and $\theta^\mu$. The critic is optimized on the mean squared error between predictions $Q(s_j, a_j; \theta^Q)$ and targets $y_j = r_j + \gamma Q^\star(s_{j+1}, \mu^\star(s_{j+1}; \theta^\mu))$, where $Q^\mu$ and $\mu^\star$ are target networks, parameterized by $\theta^Q$ and $\theta^\mu$. The parameters of $\mu$ are optimized following the deterministic policy gradient theorem (Silver et al., 2014), i.e. $\nabla_{\theta^\mu} \leftarrow \frac{\gamma}{\gamma - 1} \sum_j \nabla_{a} Q(s, a; \theta^Q)\big|_{s=s_j, a=\mu(s_j; \theta^\mu)} \nabla_{\theta^\mu} \mu(s_j; \theta^\mu)$, and the parameters of the target networks are updated according to $\theta^Q \leftarrow (1-\tau)\theta^Q + \tau Q^\star$ and $\theta^\mu \leftarrow (1-\tau)\theta^\mu + \tau \theta^\mu$, with $\tau \in [0,1]$. TD3 (Fujimoto et al., 2018) adds three adjustments to vanilla DDPG. First, the minimum prediction of two distinct critics is taken for target calculation to alleviate overestimation bias, an approach belonging to the family of Double Q-learning algorithms (van Hasselt et al., 2016). Second, Gaussian smoothing is applied to the target-policy, addressing the variance in updates. Third, actor and target networks are updated every $d$-th gradient step of the critic, to account for the problem of moving targets.

4 Off-policy Multi-step Q-Learning

In this section, we introduce the Composite Q-learning algorithm, an off-policy multi-step reinforcement learning method to enhance data-efficiency, along with MVE-TD3 and TD3(Δ) as baselines.
In order to formalize the off-policy estimation of approach we call Composite Q-learning Q-functions which we then combine to the full return with model-free Shifted Q-functions, an approach that comes naturally to achieve a certain generalization error is exponential in the horizon of the MDP. As discussed in Jin et al. (2018), Q-learning with UCB-exploration has total regret polynomial in the horizon of the MDP. As Murphy (2005) shows for the fixed-batch fitted Q-iteration case, the number of samples needed to achieve a certain generalization error is exponential in the horizon of the MDP. Under review as a conference paper at ICLR 2020

The main motivation behind this work is the assumption that learning values on short time scales can be achieved faster than for the full temporal horizon of a task, which can be prohibitively long. As Murphy [2005] shows for the fixed-batch fitted Q-iteration case, the number of samples needed to achieve a certain generalization error is exponential in the horizon of the MDP.

As Murphy (2005) shows for the fixed-batch fitted Q-iteration case, the number of samples needed to achieve a certain generalization error is exponential in the horizon of the MDP. As discussed in Jin et al. (2018), Q-learning with UCB-exploration has total regret polynomial in the horizon of the MDP and exponential with epsilon-greedy exploration (Kearns and Singh, 2002). We thus argue that truncated horizons for a fixed MDP translate to lower sample complexity of value-estimation. Building upon this idea, we estimate the return of n-step rollouts of the target-policy via Truncated Q-functions which we then combine to the full return with model-free Shifted Q-functions, an approach we call Composite Q-learning, while remaining purely off-policy. Since these quantities cannot be estimated directly from single-step transitions, we introduce a consecutive bootstrapping scheme based on intermediate predictions. For an overview, see Figure 1a. The full algorithm is in the appendix. Code can be found in the supplementary.

4.1 COMPOSITE Q-LEARNING

The main motivation behind this work is the assumption that learning values on short time scales can be achieved faster than for the full temporal horizon of a task, which can be prohibitively long. As Murphy [2005] shows for the fixed-batch fitted Q-iteration case, the number of samples needed to achieve a certain generalization error is exponential in the horizon of the MDP. As discussed in Jin et al. (2018), Q-learning with UCB-exploration has total regret polynomial in the horizon of the MDP and exponential with epsilon-greedy exploration (Kearns and Singh, 2002). We thus argue that truncated horizons for a fixed MDP translate to lower sample complexity of value-estimation. Building upon this idea, we estimate the return of n-step rollouts of the target-policy via Truncated Q-functions which we then combine to the full return with model-free Shifted Q-functions, an approach we call Composite Q-learning, while remaining purely off-policy. Since these quantities cannot be estimated directly from single-step transitions, we introduce a consecutive bootstrapping scheme based on intermediate predictions. For an overview, see Figure 1a. The full algorithm is in the appendix. Code can be found in the supplementary.

4.1.1 TRUNCATED Q-FUNCTIONS

In order to formalize the off-policy estimation of n-step returns, assume that n ≪ (T − 1) and that (T − 1 − t) mod n = 0 for task horizon T. We make use of the following observation:

\[ Q^T(s_t, a_t) = E_{t, \pi, M}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots + \gamma^{T-1} r_{T-1}] \]

\[ = E_{t, \pi, M}\left[ \sum_{j=t}^{t+n-1} \gamma^{j-t} r_j + \gamma^n \left( \sum_{j=t+n}^{t+2n-1} \gamma^{j-(t+n)} r_j \right) \right. \]

\[ + \cdots + \gamma^{T-n} \left( \sum_{j=T-n}^{T-1} \gamma^{j-(T-n)} r_j \right) \].

That is, we can define the action-value as the combination of partial sums of length n. We can then define the Truncated Q-function as

\[ Q^T_n(s_t, a_t) = E_{t, \pi, M}[\sum_{j=t}^{t+n-1} \gamma^{j-t} r_j], \]

which we plug into Equation 1:

\[ Q^T_n(s_t, a_t) = E_{t, \pi, M}[Q^T_n(s_{t+n}, a_{t+n}) + \gamma^{n} Q^T_n(s_{t+n+1}, a_{t+n+1}) + \cdots + \gamma^{T-n} Q^T_n(s_{T-n}, a_{T-n})]. \] (2)

**Theorem 1.** Let \( Q^T_n(s_t, a_t) = r_t \) be the one-step Truncated Q-function and \( Q^T_{n+1}(s_t, a_t) = r_t + \gamma E_{t, \pi, M}[Q^T_{n+1}(s_{t+1}, a_{t+1})] \) the \( n \)-step Truncated Q-function. Then \( Q^T_n(s_t, a_t) \) represents the truncated return \( Q^T_n(s_t, a_t) \).

Following Theorem 1 (the proof can be found in the supplementary), we approximate \( Q^T_n(s_t, a_t) \) off-policy via consecutive bootstrapping. Let \( Q^T(\cdot, \cdot; \theta^Q) \) denote a function approximator with
parameters $\theta Q_n$ and $n$ outputs, subsequently called heads, estimating $Q^r_s$. Each output $Q^r_s$ bootstraps from the prediction of the preceding head, with the first approximating the immediate reward function. The targets are therefore given by:

$$y^r_{j,1} = r_j \text{ and } y^r_{j,i>1} = r_j + \gamma Q^r_{i-1}(s_{j+1},\mu(s_j+1)\theta^r)|\theta^r_{Q_{i-1}}), \quad (3)$$

where $\mu'$ corresponds to the actor maximizing the full Q-value as defined in Section 3. That is, $Q^r$ represents evaluations of $\mu$ at different stages of truncation and $y^r_{j,i<n}$ serve as intermediate predictions to get $y^r_{j,n}$. We then only use $Q^r_n$, which implements the full $n$-step return, as the first part of the composition of the Q-target. Please note that in order to estimate Equation (2), the dynamics model would be needed to get $s_{t+c}$ of a rollout starting in $s_t$. In the following, we describe an approach to achieve an estimation of Equation (2) model-free.

### 4.1.2 Shifted Q-functions

To get an estimation for the remainder of the rollout $Q^\pi_{n,\infty} = E_{t,\pi,M}[\gamma^n Q(s_{t+n},a_{t+n})]$ after $n$ steps, we use a consecutive bootstrapping formulation of the Q-prediction.

**Theorem 2.** Let $Q^\pi_{n,n}(s_t,a_t) = E_{t,\pi,M}[\gamma^n Q(s_{t+n},a_{t+n})]$ be the one-step Shifted Q-function and $Q^\pi_{i+1,\infty}(s_t,a_t) = E_{t,\pi,M}[\gamma^{i} Q(s_{t+i+1},a_{t+i+1})]$ the $i$-step Shifted Q-function. Then $Q^\pi_{n,\infty}(s_t,a_t) = E_{t,\pi,M}[\gamma^i Q_i(s_{t+i},a_{t+i})]$.

Again, the proof of Theorem 2 can be found in the supplementary. Let $Q^\text{Sh}(\cdot,\cdot;\theta^Q_n)$ denote the function approximator estimating the Shifted Q-function $Q^\pi_{n,\infty}$, parameterized by $\theta^Q_n$. We can shift the Q-prediction by bootstrapping without taking the immediate reward into account, so as to skip the first $n$ rewards of a target-policy rollout. The Shifted Q-targets for heads $Q^\text{Sh}$ therefore become:

$$y^\text{Sh}_{j,1} = \gamma Q^r(s_{j+1},\mu(s_j+1)\theta^r)|\theta^Q_n) \text{ and } y^\text{Sh}_{j,i>1} = \gamma Q^\text{Sh}_{i-1}(s_{j+1},\mu(s_j+1)\theta^r)|\theta^Q_{n,i-1}). \quad (4)$$

### 4.1.3 Composition

Following the definitions of Truncated and Shifted Q-functions, we can compose the full return.

**Theorem 3.** Let $Q^\pi_{n,n}(s_t,a_t) = E_{t,\pi,M}[\gamma^n Q(s_{t+n},a_{t+n})]$ be the truncated return and $Q^\pi_{n,\infty}(s_t,a_t) = E_{t,\pi,M}[\gamma^n Q(s_{t+n},a_{t+n})]$ the shifted return. Then $Q^\pi_{n,\infty}(s_t,a_t) = Q^\pi_n(s_t,a_t) + Q^\pi_{n,\infty}(s_t,a_t)$ represents the full return, i.e. $Q^\pi_n(s_t,a_t) = E_{t,\pi,M}[\gamma^n Q(s_{t+n},a_{t+n})]$.

The incorporation of truncated returns breaks down the time scale of the long-term prediction by the Shifted Q-function. For details, see the proof of Theorem 3 in the supplementary. We can thus form the Composite Q-target as:

$$y^Q_j = r_j + \gamma(Q^r_{j+1},\mu(s_{j+1})|\theta^r)|\theta^Q_n) + Q^\text{Sh}_{n}(s_{j+1},\mu(s_j+1)|\theta^r)|\theta^Q_n) \quad (5)$$

approximated by $Q^C(\cdot,\cdot;\theta^Q_n)$ with parameters $\theta^Q_n$. Since we have true reward $r_j$, we include it in the target. We jointly estimate $Q^r, Q^\text{Sh}$ and $Q$ with function approximator $Q^C(\cdot,\cdot;\theta^Q_n)$. Let $SE_c$ denote the square error between targets $y^Q_j$ and predictions $Q^C(s_{j+n},a_j|\theta^Q_n)$. See Figure 1b for the detailed architecture of the Composite Q-network. Each pair $Q^r_j + Q^\text{Sh}_j, 1 \leq j \leq n$, is a complete approximation of the true Q-value. The circular dependency can lead to stability issues, due to the amplification of propagated errors. We add a regularization term to the loss penalizing the deviation between the prediction of $Q$ and the $n$ different Q-pairs to keep estimates in a narrow range. It is implemented as the mean squared error, i.e. the loss function becomes:

$$L = \frac{1}{m} \sum_{j=1}^{m} \left( SE_c + \beta \frac{1}{n} \sum_{i=1}^{n} \left( Q(s_j,a_j|\theta^Q_n) - (Q^r_{i}(s_{j+1},a_j|\theta^Q_n) + Q^\text{Sh}_i(s_{j+1},a_j|\theta^Q_n) \right)^2 \right), \quad (6)$$

for batch of size $m$ and with $\beta$ being the regularization weight. Actor $\mu$ is then updated on $Q$.

### 4.2 TD3($\Delta$) and MVE-TD3

Most related to Composite Q-learning are TD($\Delta$) [Romoff et al., 2019] and MVE-DDPG [Feinberg et al., 2018]. In this section, we describe how to extend TD($\Delta$) to an off-policy setting and how to combine MVE and TD3.
**TD3(Δ)** Another way to divide the value function into multiple time scales is TD(Δ) [Romoff et al., 2019]. To this point, it has only been applied in an on-policy setting. In favor of comparability, we extend TD(Δ) to Q-learning, yielding TD3(Δ). The main idea of TD(Δ) is the combination of different value functions corresponding to increasing discount values. Let \( \gamma^\Delta \) denote a fixed ordered sequence of increasing discount values, i.e. \( \gamma^\Delta = (\gamma_1, \gamma_2, \ldots, \gamma_k)^T \). We can then define delta functions \( W_i \) as \( W_1 = Q_{\gamma_1} \) and \( W_{i+1} = Q_{\gamma_i} - Q_{\gamma_{i-1}} \). Let \( Q^\Delta(\cdot; \theta^Q) \) denote the function approximator estimating \( Q_{\gamma_{i-1}} \). Based on the derivations in [Romoff et al., 2019], the targets for Q-learning can be formalized as:

\[
y^\gamma_{j,1} = r_j + \gamma Q'_{\gamma_1}(s_{j+1}, \mu'(s_{j+1}|\theta^\mu))|\theta^Q_{\gamma_1} \quad \text{and} \quad y^\gamma_{j,i+1} = (\gamma_i - \gamma_{i-1}) Q'_{\gamma_{i-1}}(s_{j+1}, \mu'(s_{j+1}|\theta^\mu))|\theta^Q_{\gamma_{i-1}} + \gamma_i W_i(s_{j+1}, \mu'(s_{j+1}|\theta^\mu)|\theta^{W_i}),
\]

which can then be used in any Q-learning algorithm. The authors suggest the use of \( n \)-step targets within TD(Δ) which is not easily applicable in an off-policy setting. In our experiments, we therefore compare our approach to single-step TD3(Δ). The algorithm can be found in the appendix.

**MVE-TD3** We also apply Model-based Value Expansion within TD3, subsequently called MVE-TD3. We add Gaussian policy smoothing to the rollout of the model. In contrast to [Feinberg et al., 2018], however, we do not assume to have knowledge about the reward function. Our model therefore approximates both, dynamics and reward.

5 EXPERIMENTAL RESULTS

We evaluate Composite Q-learning in both, the tabular setting and the actor-critic method TD3.

5.1 Tabular Composite Q-learning

To analyze the effect of incorporating short-term prediction \( Q_{n}^{\text{Tr}} \) in the Q-update, we apply Composite Q-learning in the tabular case to the MDP of horizon \( K \) given in Figure 2a. We compare it to vanilla Q-learning, as well as multi-step Q-learning based on subtrajectories of the exploratory policy and imaginary rollouts of the target-policy with the true model of the MDP.

![Figure 2](image-url)
Results for \( K = 20 \) are depicted in Figure 2b. All approaches update the Q-function with a learning rate of \( 10^{-3} \) on the same fixed batch of \( 10^3 \) episodes with a percentage of 10\% non-optimal transitions. For the multi-step approaches, we set rollout length \( n = 4 \) and update the Shifted Q-function with a learning rate of \( 10^{-2} \). The short-term predictions converge much faster than the value function in Q-learning. Based on the results in Figure 2b, we believe this to be also the reason for the folded learning curve of Composite Q-learning. In the beginning, the target-policy is non-optimal, hence the Truncated Q-functions adapt to the non-optimal policy rather quickly. However, once the truncated estimations represent the optimal values, they are propagated to the full return in Composite Q-learning almost immediately. The erroneous updates of on-policy multi-step Q-learning lead to convergence to a wrong action-value which is underlining the importance of truly off-policy learning. However, if the true model can be used for the rollout, this can be really effective. The difference in convergence speed between Q-learning and Composite Q-learning grows with increasing horizon, as shown in Table 1. This is in line with the findings of Jin et al. (2018) who establish a connection between the horizon of an MDP and the sample complexity of value estimation.

Table 1: Comparison of convergence speed between tabular Q-learning and tabular Composite Q-learning for the MDP given in Figure 2a with \( n = 4 \).

<table>
<thead>
<tr>
<th>Horizon ( K )</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed up to Q-learning</td>
<td>11.13%</td>
<td>44.28%</td>
<td>57.47%</td>
<td>66.16%</td>
</tr>
</tbody>
</table>

5.2 Composite Q-Learning with Function Approximation

We evaluate TD3, Composite TD3, MVE-TD3 and TD3(\( \Delta \)) on three robot simulation tasks of OpenAI Gym (Brockman et al., 2016) based on MuJoCo (Todorov et al., 2012): Walker2d-v2, Ant-v2 and Hopper-v2. A visualization of the environments is depicted in Figure 3.

Figure 3: Visualization of Walker2d-v2 (left), Ant-v2 (middle) and Hopper-v2 (right).

Parameter Setting Since the underlying algorithm is TD3 in all cases, we keep the main parameters the same across all approaches. Learning rate, target update and actor setting are the same as in the default setting of TD3. We use Gaussian exploration noise with \( \sigma = 0.15 \). The critic in Composite Q-learning consists of four layers with 500 neurons and leaky ReLU activation, the critic in all other approaches and the model in MVE-TD3, with a learning rate of \( 10^{-3} \), has two layers. However, we also evaluate TD3 and TD3(\( \Delta \)) with the same critic setting as Composite TD3, see Figure 6 in the supplementary. This corresponds to the most stable setting we could find for TD3 with the given subset of MuJoCo tasks. For Composite TD3, we set \( n = 50 \) and \( \beta = 10^{-4} \) for Walker2d-v2 and Hopper-v2 and \( \beta = 5 \cdot 10^{-5} \) for Ant-v2. For TD3(\( \Delta \)), we use the \( \gamma \)-schedule as suggested by the authors, i.e. \( \gamma_1 = 0 \) and \( \gamma_{i>1} = \frac{\gamma_i - 1}{2} \), with an upper limit of 0.99 (Romoff et al., 2019). For MVE-TD3, we use a rollout length of 3, as described in Feinberg et al. (2018).

Data Efficiency We choose the area under the learning curve as performance measure, which is a common way to evaluate data-efficiency and learning stability (see e.g. Hessel et al. (2018)). Since vanilla TD3 can have severe stability issues for a small number of runs, we compare median and interquartile ranges. As listed in Table 2, Composite TD3 outperforms TD3, as well as state-of-the-art multi-step approaches, in terms of learning speed throughout training. After the considered time frame of \( 4 \cdot 10^5 \) transitions, Composite TD3 has a 18\%, 11\% and 19\% larger area under the median learning curve, in comparison to vanilla TD3. As depicted in Figure 4, MVE-TD3 is highly sensitive w.r.t. accumulating errors in reward and state prediction, even when applying the TD-\( k \) trick.
Figure 4: Results (top) and TD-errors (bottom) for Walker2d-v2 (left), Ant-v2 (middle) and Hopper-v2 (right). The plots show median and interquartile ranges over 11 training runs, each representing mean evaluation performance over 100 initial states. The lower plots show TD-errors over time for the different horizons of the Truncated Q-function, as well as the TD-errors for the complete Q-estimate. Please note that \textit{TD-error} here means the deviation from the associated target.

However, if dynamics and reward are estimated with high accuracy, model-based rollouts can be very effective, as the results show for Ant-v2. Based on our experiments, TD3(\(\Delta\)) seems to be less sensitive than MVE-TD3, albeit also less efficient than Composite TD3. The maximum return found by all approaches within \(4 \cdot 10^5\) transitions can be seen in Table 3.

Table 2: Normalized area under the median learning curve over 11 training runs.

<table>
<thead>
<tr>
<th>Samples</th>
<th>Method</th>
<th>Walker2d-v2</th>
<th>Ant-v2</th>
<th>Hopper-v2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2 \cdot 10^5)</td>
<td>TD3</td>
<td>81.63%</td>
<td>87.73%</td>
<td>84.02%</td>
</tr>
<tr>
<td></td>
<td>Composite TD3</td>
<td>\textbf{100.00%}</td>
<td>\textbf{100.00%}</td>
<td>\textbf{100.00%}</td>
</tr>
<tr>
<td></td>
<td>MVE-TD3</td>
<td>49.71%</td>
<td>81.56%</td>
<td>86.02%</td>
</tr>
<tr>
<td></td>
<td>TD3((\Delta))</td>
<td>89.04%</td>
<td>96.79%</td>
<td>73.85%</td>
</tr>
<tr>
<td>(3 \cdot 10^5)</td>
<td>TD3</td>
<td>78.97%</td>
<td>87.64%</td>
<td>78.00%</td>
</tr>
<tr>
<td></td>
<td>Composite TD3</td>
<td>\textbf{100.00%}</td>
<td>\textbf{100.00%}</td>
<td>\textbf{100.00%}</td>
</tr>
<tr>
<td></td>
<td>MVE-TD3</td>
<td>44.27%</td>
<td>85.92%</td>
<td>56.62%</td>
</tr>
<tr>
<td></td>
<td>TD3((\Delta))</td>
<td>86.65%</td>
<td>94.35%</td>
<td>71.80%</td>
</tr>
<tr>
<td>(4 \cdot 10^5)</td>
<td>TD3</td>
<td>81.25%</td>
<td>88.93%</td>
<td>80.68%</td>
</tr>
<tr>
<td></td>
<td>Composite TD3</td>
<td>\textbf{100.00%}</td>
<td>\textbf{100.00%}</td>
<td>\textbf{100.00%}</td>
</tr>
<tr>
<td></td>
<td>MVE-TD3</td>
<td>43.94%</td>
<td>88.03%</td>
<td>43.77%</td>
</tr>
<tr>
<td></td>
<td>TD3((\Delta))</td>
<td>87.98%</td>
<td>94.50%</td>
<td>79.93%</td>
</tr>
</tbody>
</table>
Under review as a conference paper at ICLR 2020

Table 3: Maximum return of the median learning curve over 11 training runs within $4 \cdot 10^5$ samples.

<table>
<thead>
<tr>
<th>Environment</th>
<th>TD3</th>
<th>Composite TD3</th>
<th>MVE-TD3</th>
<th>TD3(\Delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walker2d-v2</td>
<td>3961.79</td>
<td><strong>4390.80</strong></td>
<td>2049.99</td>
<td>4101.27</td>
</tr>
<tr>
<td>Ant-v2</td>
<td>2933.26</td>
<td><strong>3172.28</strong></td>
<td>3019.21</td>
<td>3042.50</td>
</tr>
<tr>
<td>Hopper-v2</td>
<td>2888.24</td>
<td>3043.91</td>
<td>1080.72</td>
<td><strong>3046.59</strong></td>
</tr>
</tbody>
</table>

**TD-error Analysis** To test our intuition stated in the motivation, we refer to the lower row of Figure 4 which shows the TD-errors for the different stages of truncation, as well as the TD-errors for the complete Q-estimate. All Truncated Q-estimates have lower TD-error throughout learning and it can be seen that the TD-error is consistently higher for longer horizons. This reaffirms our hypothesis that learning is easier on shorter horizons. The Composite Q-target therefore reflects the true action-value faster which is beneficial for Q-learning.

**Sensitivity to Hyperparameters** Lastly, we evaluate the influence of $n$ and $\beta$, exemplary for the Walker2d-v2 environment. The results can be seen in Figure 5. Shorter truncation horizons can lead to faster convergence, but the variance increases. If $n$ is rather large, however, it can slow learning down. The same holds if the regularization weight $\beta$ is set to a high value. On the other hand, regularization is needed to keep the predictions in a narrow range, since it can lead to stability issues, otherwise.

6 CONCLUSION

We introduced Composite Q-learning, an off-policy learning method that divides the long-term value into smaller time scales. It combines Truncated Q-functions acting on a short horizon with Shifted Q-functions for the remainder of the rollout. We analyzed the efficacy of Composite Q-learning in the tabular case and showed that the benefit of short-term predictions increases with growing task horizon. We further evaluated MVE-TD3 and introduced TD3(\Delta), an off-policy variant of TD(\Delta). We showed on three simulated robot tasks that Composite TD3 outperforms vanilla TD3 by 18%, 11% and 19% in terms of area under the median learning curve. We also showed that Composite TD3 is able to achieve state-of-the-art data-efficiency compared to other approaches in off-policy multi-step learning.

Going forward, the uncertainty estimate based on the variance of the Composite Q-network could be of benefit in both, update calculation and exploration. We further leave the application of the truncated formulation of other quantities such as state change or auxiliary costs as future work.
REFERENCES


A Algorithm for Composite TD3

A detailed description of Composite DDPG is given in Algorithm 1 where the adjustments of TD3 are omitted for simplicity. In order to transform Algorithm 1 to its TD3-equivalent, Gaussian policy smoothing has to be added to all targets in Line 8, as well as taking the minimum prediction of two distinct critics for each target. Furthermore, actor and target networks have to be updated with delay.

**Algorithm 1: Composite DDPG**

1. initialize critic $Q^c$, actor $\mu$ and targets $Q^{c^t}$, $\mu'$
2. initialize replay buffer $\mathcal{R}$
3. for episode = 1..E do
4.   get initial state $s_1$
5.   for $t = 1..T$ do
6.     apply action $a_t = \mu(s_t|\theta^\mu) + \xi$, where $\xi \sim \mathcal{N}(0, \sigma)$
7.     observe $s_{t+1}$ and $r_t$ and save transition $(s_t, a_t, s_{t+1}, r_t)$ in $\mathcal{R}$
8.     calculate targets:
9.       \[ y_{t|1}^{\mu} = r_t \]
10.      \[ y_{t|>1}^{\mu} = r_t + \gamma Q^{c^t}_{t|>1}(s_{t+1}, \mu'(s_{t+1}|\theta^\mu))|Q^{c^t}_{t|>1} \]
11.      \[ y_{t|1}^{\mu} = \gamma Q^{c^t}(s_{t+1}, \mu'(s_{t+1}|\theta^\mu))|Q^{c^t} \]
12.      \[ y_{t|>1}^{\mu} = \gamma Q^{c^t}_{t|>1}(s_{t+1}, \mu'(s_{t+1}|\theta^\mu))|Q^{c^t}_{t|>1} \]
13.     apply action $\tilde{a}_t = \mu(s_t|\theta^\mu) + \tilde{\xi}$
14.     observe $s_{t+1}$ and $\tilde{r}_t$ and save transition $(s_t, \tilde{a}_t, s_{t+1}, \tilde{r}_t)$ in $\mathcal{R}$
15.     calculate targets:
16.       \[ y_{t|1}^{\mu} = \tilde{r}_t \]
17.      \[ y_{t|>1}^{\mu} = \tilde{r}_t + \gamma Q^{c^t}_{t|>1}(s_{t+1}, \mu'(s_{t+1}|\theta^\mu))|Q^{c^t}_{t|>1} \]
18.      \[ y_{t|1}^{\mu} = \gamma Q^{c^t}(s_{t+1}, \mu'(s_{t+1}|\theta^\mu))|Q^{c^t} \]
19.     \[ y_{t|>1}^{\mu} = \gamma Q^{c^t}_{t|>1}(s_{t+1}, \mu'(s_{t+1}|\theta^\mu))|Q^{c^t}_{t|>1} \]
20.     \[ y_{t}^{\mu} = \left[ y_{t|1}^{\mu}, y_{t|>1}^{\mu}, y_{t|1}^{\mu}, y_{t|>1}^{\mu} \right] \]
21.   update $Q^c$ on minibatch $b$ of size $m$ from $\mathcal{R}$ according to Equation 6
22.   update $\mu$ on $Q$
23.   adjust parameters of $Q^{c^t}$ and $\mu'$

B Algorithm for TD3($\Delta$)

A detailed description of DDPG($\Delta$) is given in Algorithm 2. The authors suggest the use of $n$-step samples which is not easily applicable in an off-policy setting. In our experiments, we therefore compare our approach to single-step TD3($\Delta$). To transform DDPG($\Delta$) to TD3($\Delta$), the adjustments as described in Appendix A have to be applied analogously.

**Algorithm 2: DDPG($\Delta$)**

1. initialize critic $Q^\Delta$, actor $\mu$ and targets $Q^{\Delta^t}$, $\mu'$
2. initialize replay buffer $\mathcal{R}$
3. set discount values $\gamma^\Delta = (\gamma_0, \gamma_1, \ldots, \gamma_k)^T$
4. for episode = 1..E do
5.   get initial state $s_1$
6.   for $t = 1..T$ do
7.     apply action $a_t = \mu(s_t|\theta^\mu) + \xi$, where $\xi \sim \mathcal{N}(0, \sigma)$
8.     observe $s_{t+1}$ and $r_t$ and save transition $(s_t, a_t, s_{t+1}, r_t)$ in $\mathcal{R}$
9.     calculate targets:
10.    \[ y_{t|1}^{\mu} = r_t + \gamma Q_{t|1}^\Delta(s_{t+1}, \mu'(s_{t+1}|\theta^\mu)|Q_{t|1}^\Delta \]
11.    \[ y_{t|>1}^{\mu} = (1 - \gamma_{t-1})Q_{t|>1}^\Delta(s_{t+1}, \mu'(s_{t+1}|\theta^\mu)|Q_{t|>1}^\Delta \]
12.    \[ + \gamma W^\mu(s_{t+1}, \mu'(s_{t+1}|\theta^\mu)|W^\mu) \]
13.     apply action $\tilde{a}_t = \mu(s_t|\theta^\mu) + \tilde{\xi}$
14.     observe $s_{t+1}$ and $\tilde{r}_t$ and save transition $(s_t, \tilde{a}_t, s_{t+1}, \tilde{r}_t)$ in $\mathcal{R}$
15.     calculate targets:
16.    \[ y_{t|1}^{\mu} = \tilde{r}_t \]
17.    \[ y_{t|>1}^{\mu} = (1 - \gamma_{t-1})Q_{t|>1}^\Delta(s_{t+1}, \mu'(s_{t+1}|\theta^\mu)|Q_{t|>1}^\Delta \]
18.    \[ + \gamma W^\mu(s_{t+1}, \mu'(s_{t+1}|\theta^\mu)|W^\mu) \]
19.     \[ y_{t}^{\mu} = \left[ y_{t|1}^{\mu}, y_{t|>1}^{\mu}, y_{t|1}^{\mu}, y_{t|>1}^{\mu} \right] \]
20.     update $Q^\Delta$ on minibatch $b$ of size $m$ from $\mathcal{R}$
21.     update $\mu$ on $Q_{t|n}$
22.     adjust parameters of $Q^{\Delta^t}$ and $\mu'$
C Truncated Q-functions

**Theorem 1.** Let \( Q^\pi_t(s_t, a_t) = r_t \) be the one-step Truncated Q-function and \( Q^\pi_{t>1}(s_t, a_t) = r_t + \gamma E_{t, \pi, \mathcal{M}}[Q^\pi_{t-1}(s_{t+1}, a_{t+1})] \) the \( i \)-step Truncated Q-function. Then \( Q^\pi_t(s_t, a_t) \) represents the truncated return \( Q^\pi_{t+1}(s_t, a_t) = E_{t, \pi, \mathcal{M}}[\sum_{j=t}^{t+i-1} \gamma^{j-t}r_j] \).

**Proof.** Proof by induction. \( Q^\pi_t(s_t, a_t) = r_t \) by definition. The theorem follows from induction step:

\[
Q^\pi_t(s_t, a_t) = r_t + \gamma E_{t, \pi, \mathcal{M}}[Q^\pi_{t-1}(s_{t+1}, a_{t+1})] \\
= r_t + \gamma E_{t, \pi, \mathcal{M}} \left[ \sum_{j=(t+1)}^{(t+1)+(i-1)-1} \gamma^{j-(t+1)}r_j \right] \\
= r_t + \gamma E_{t, \pi, \mathcal{M}} \left[ \sum_{j=(t+1)}^{t+i-1} \gamma^{j-(t+1)}r_j \right] \\
= r_t + E_{t, \pi, \mathcal{M}} \left[ \sum_{j=(t+1)}^{t+i-1} \gamma^{j-t}r_j \right] \\
= E_{t, \pi, \mathcal{M}} \left[ \sum_{j=t}^{t+i-1} \gamma^{j-t}r_j \right].
\]

\( \square \)

D Shifted Q-functions

**Theorem 2.** Let \( Q^\pi_{t<:\infty}(s_t, a_t) = E_{t, \pi, \mathcal{M}}[\gamma Q^\pi_t(s_{t+1}, a_{t+1})] \) be the one-step Shifted Q-function and \( Q^\pi_{t>1:\infty}(s_t, a_t) = E_{t, \pi, \mathcal{M}}[\gamma Q^\pi_{t-1:\infty}(s_{t+1}, a_{t+1})] \) the \( i \)-step Shifted Q-function. Then \( Q^\pi_{t<:\infty}(s_t, a_t) \) represents the shifted return \( Q^\pi_{t>1:\infty}(s_t, a_t) = E_{t, \pi, \mathcal{M}}[\gamma^i Q^\pi_t(s_{t+i}, a_{t+i})] \).

**Proof.** Proof by induction. \( Q^\pi_{t<:\infty}(s_t, a_t) = E_{t, \pi, \mathcal{M}}[\gamma Q^\pi_t(s_{t+1}, a_{t+1})] \) by definition. The theorem follows from induction step:

\[
Q^\pi_{t<:\infty}(s_t, a_t) = E_{t, \pi, \mathcal{M}} \left[ \gamma Q^\pi_{t-1<:\infty}(s_{t+1}, a_{t+1}) \right] \\
= E_{t, \pi, \mathcal{M}} \left[ \gamma(\gamma Q^\pi_{t-1<:\infty}(s_{t+1+i-1}, a_{t+1+i-1})) \right] \\
= E_{t, \pi, \mathcal{M}} \left[ \gamma(\gamma Q^\pi_{t-1<:\infty}(s_{t+i}, a_{t+i})) \right] \\
= E_{t, \pi, \mathcal{M}} \left[ \gamma^i Q^\pi_t(s_{t+i}, a_{t+i}) \right].
\]

\( \square \)

E Resolving the Mutual Recursion

**Theorem 3.** Let \( Q^\pi_{n<:\infty}(s_t, a_t) = E_{t, \pi, \mathcal{M}}[\sum_{j=t}^{t+n-1} \gamma^{j-t}r_j] \) be the truncated return and \( Q^\pi_{n>1:\infty}(s_t, a_t) = E_{t, \pi, \mathcal{M}}[\gamma^n Q^\pi_t(s_{t+n}, a_{t+n})] \) the shifted return. Then \( Q^\pi_t(s_t, a_t) = Q^\pi_n(s_t, a_t) + Q^\pi_{n<:\infty}(s_t, a_t) \) represents the full return, i.e. \( Q^\pi_t(s_t, a_t) = E_{t, \pi, \mathcal{M}}[\sum_{j=t}^{\infty} \gamma^{j-t}r_j] \).
Proof.

\[ Q^\pi(s_t, a_t) = Q^\pi_n(s_t, a_t) + Q^\pi_{n:\infty}(s_t, a_t) \]

\[ = E_{t,\pi,M} \left[ \sum_{j=t}^{t+n-1} \gamma^{j-t} r_j + \gamma^n Q^\pi_n(s_{t+n}, a_{t+n}) \right] \]

\[ = E_{t,\pi,M} \left[ \sum_{j=t}^{t+n-1} \gamma^{j-t} r_j + \gamma^n \left( \sum_{j=t+n}^{t+2n-1} \gamma^{j-t-n} r_j + \gamma^n Q^\pi(s_{t+2n}, a_{t+2n}) \right) \right] \]

\[ = E_{t,\pi,M} \left[ \sum_{j=t}^{t+2n-1} \gamma^{j-t} r_j + \gamma^{2n} Q^\pi(s_{t+2n}, a_{t+2n}) \right] \]

Repeating this process then gives \( Q^\pi(s_t, a_t) = E_{t,\pi,M} \left[ \sum_{j=t}^{\infty} \gamma^{j-t} r_j \right] \).

Due to the composite structure, the Shifted Q-function represents the long-term sum of partial returns provided by the Truncated Q-function, as opposed to single reward values.

\section*{F TD3 and TD3(\Delta) with Larger Networks}

To further analyze the impact of the Composite Q-learning structure, we evaluate TD3 and TD3(\Delta) with the same number of parameters for the critic, as depicted in Figure 6. The approaches do not seem to make use of the additional capacity and reveal a worse performance in the given time frame.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure6.png}
\caption{Results for TD3 and TD3(\Delta) with the same number of parameters for the critic as Composite TD3, i.e. four layers with 500 neurons. The plots show median and interquartile ranges over 11 training runs, each representing mean evaluation performance over 100 initial states. We denote approaches using a critic with four layers by 4L.}
\end{figure}