Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)
Sister Conferences Best Papers Track

Deep Residual Reinforcement Learning (Extended Abstract)

Shangtong Zhang* , **Wendelin Boehmer** , **Shimon Whiteson**
University of Oxford
{shangtong.zhang, wendelin.boehmer, shimon.whiteson}@cs.ox.ac.uk

Abstract

We revisit *residual algorithms* in both model-free and model-based reinforcement learning settings. We propose the *bidirectional target network* technique to stabilize residual algorithms, yielding a residual version of DDPG that significantly outperforms vanilla DDPG in commonly used benchmarks. Moreover, we find the residual algorithm an effective approach to the *distribution mismatch* problem in model-based planning. Compared with the existing TD(k) method, our residual-based method makes weaker assumptions about the model and yields a greater performance boost.

1 Introduction

Semi-gradient algorithms have recently enjoyed great success in deep reinforcement learning (RL) problems, e.g., DQN [Mnih *et al.*, 2015] achieves human-level control in the Arcade Learning Environment (ALE, [Bellemare *et al.*, 2013]). However, such algorithms lack theoretical support. Most semi-gradient algorithms suffer from divergence under nonlinear function approximation or off-policy training [Baird, 1995; Tsitsiklis and Van Roy, 1997]. By contrast, *residual gradient* (RG, [Baird, 1995]) algorithms are true stochastic gradient algorithms and enjoy convergence guarantees (to a local minimum) under mild conditions with both nonlinear function approximation and off-policy training. [Baird, 1995] further proposes *residual algorithms* (RA) to unify residual gradients and semi-gradients via mixing them together.

Residual algorithms, however, suffer from the double sampling issue [Baird, 1995]: two independently sampled successor states are required to compute the residual gradients. This requirement can be easily satisfied in model-based RL or in deterministic environments. However, even in these settings, residual algorithms have long been either overlooked or dismissed as impractical. In this paper, we aim to overturn that conventional wisdom with new algorithms built on RA and empirical results showing their efficacy.

Our contributions are twofold. First, we show the advantages of RA in a model-free RL setting with deterministic environments. While target networks [Mnih *et al.*, 2015] are usually an important component in deep RL algorithms to stabilize training [Mnih *et al.*, 2015; Lillicrap *et al.*, 2015], we find a naive combination of target networks and residual algorithms, in general, does not improve performance. Therefore, we propose the *bidirectional target network* technique to stabilize residual algorithms. We show that our residual version of Deep Deterministic Policy Gradients (DDPG, [Lillicrap *et al.*, 2015]) significantly outperforms vanilla DDPG in the DeepMind Control Suite (DMControl, [Tassa *et al.*, 2018]) and MuJoCo benchmarks.

Second, we show the advantages of RA in a model-based RL setting, where a learned model generates imaginary transitions to train the value function. In general, model-based methods suffer from a *distribution mismatch* problem [Feinberg *et al.*, 2018]. The value function trained on real states does not generalize well to imaginary states generated by a model. To address this issue, [Feinberg *et al.*, 2018] train the value function on both real and imaginary states via the TD(k) trick. However, TD(k) requires that predictions k steps in the future made by model rollouts will be accurate [Feinberg *et al.*, 2018]. In this paper, we show that RA naturally allows the value function to be trained on both real and imaginary states and requires only 1-step rollouts. Our experiments show that RA-based planning boosts performance more than TD(k)-based planning in most cases.

2 Background

We consider an MDP [Puterman, 2014] consisting of a finite state space $\mathcal{S}$, a finite action space $\mathcal{A}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a transition kernel $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ and a discount factor $\gamma \in [0, 1)$. With $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ denoting a policy, at time t, an agent at a state S_t takes an action A_t according to $\pi(\cdot|S_t)$. The agent then gets a reward R_{t+1} satisfying $\mathbb{E}[R_{t+1}] = r(S_t, A_t)$ and proceeds to a new state S_{t+1} according to $p(\cdot|S_t, A_t)$. We use $G_t \doteq \sum_{i=t+1}^{\infty} \gamma^{i-t-1} R_i$ to denote the return from time t, $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$ to denote the state value function of π, and $q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$ to denote the state-action value function of π. In the rest of this section, we use a bold capital letter to denote a matrix and a bold lowercase letter to denote a column vector. We use $\mathbf{P}_\pi$ to denote the transition matrix induced by a policy π, i.e., $\mathbf{P}_\pi[s, s'] \doteq \sum_a \pi(s, a) p(s'|s, a)$, and use d_π to denote its unique stationary distribution, assuming the

*Contact Author

chain induced by $\mathbf{P}_\pi$ is ergodic. The reward vector induced by π is $\mathbf{r}_\pi[s] = \sum_a \pi(a|s) r(s,a)$.

The value function v_π is the unique fixed point of the Bellman operator $\mathcal{T}$ [Bellman, 1957]. In a matrix form, $\mathcal{T}$ is defined as $\mathcal{T}\mathbf{v} \doteq \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}$, where $\mathbf{v}$ can be any vector in $\mathbb{R}^N$. Here $N \doteq |\mathcal{S}|$ is the number of states.

Policy Evaluation: We consider the problem of finding v_π for a given policy π and use $\mathbf{v}$, parameterized by $\mathbf{w} \in \mathbb{R}^d$, to denote an estimate of $\mathbf{v}_\pi$, the vector form of v_π. We start with on-policy linear function approximation and use $x : \mathcal{S} \to \mathbb{R}^d$ to denote a feature function which maps a state to a d-dimensional feature. The feature matrix is then $\mathbf{X} \doteq [\mathbf{x}(s_1), \dots, \mathbf{x}(s_N)]^\mathrm{T} \in \mathbb{R}^{N \times d}$, and the value estimate is $\mathbf{v} \doteq \mathbf{X}\mathbf{w}$. Two commonly used objectives for learning $\mathbf{w}$ are the Mean Squared Projected Bellman Error (MSPBE) and the Mean Squared Bellman Error (MSBE):

$$\text{MSPBE}(\mathbf{w}) \doteq ||\mathbf{v} - \Pi\mathcal{T}\mathbf{v}||^2_{d_\pi}, \quad \text{MSBE}(\mathbf{w}) \doteq ||\mathbf{v} - \mathcal{T}\mathbf{v}||^2_{d_\pi}.$$

Here Π is a projection operator which maps an arbitrary vector onto the column vector space of $\mathbf{X}$, minimizing a d_π-weighted projection error, i.e., $\Pi\mathbf{v} \doteq \mathbf{X}\bar{\mathbf{w}}$, where $\bar{\mathbf{w}} \doteq \arg\min_{\mathbf{w}} ||\mathbf{v} - \mathbf{X}\mathbf{w}||^2_{d_\pi}$. With linear function approximation and fixed features, Π is linear.

There are various algorithms for minimizing MSPBE and MSBE. Temporal Difference learning (TD, [Sutton, 1988]) is commonly used to minimize MSPBE. TD updates $\mathbf{w}$ as

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\big(R_{t+1} + \gamma \mathbf{v}(S_{t+1}) - \mathbf{v}(S_t)\big)\nabla_{\mathbf{w}} \mathbf{v}(S_t),$$

where α is a step size. Under mild conditions, on-policy linear TD converges to the point where MSPBE is 0 [Tsitsiklis and Van Roy, 1997]. TD is a *semi-gradient* [Sutton and Barto, 2018] algorithm in that it ignores the dependency of $\mathbf{v}(S_{t+1})$ on $\mathbf{w}$. There are also *true gradient* algorithms for optimizing MSPBE, e.g., Gradient TD methods [Sutton *et al.*, 2009]. Gradient TD methods compute the gradient of MSPBE directly and also enjoy convergence guarantees.

[Baird, 1995] proposes *residual gradient* algorithms for minimizing MSBE, which updates $\mathbf{w}$ as

$$\begin{aligned}\mathbf{w} \leftarrow \mathbf{w} - \alpha\big(&R_{t+1} + \gamma \mathbf{v}(S_{t+1}) - \mathbf{v}(S_t)\big) \\ &\cdot\big(\gamma \nabla_{\mathbf{w}} \mathbf{v}(S'_{t+1}) - \nabla_{\mathbf{w}} \mathbf{v}(S_t)\big),\end{aligned} \tag{1}$$

where S'_{t+1} is another sampled successor state for S_t, independent of S_{t+1}. This requirement for two independent samples is known as the *double sampling issue* [Baird, 1995]. If both the transition kernel p and the policy π are deterministic, we can simply use one sample without introducing bias. Otherwise, we may need to have access to the transition kernel p, which is usually not available in model-free RL. Regardless, RG is a true gradient algorithm with convergence guarantees under mild conditions.

We now expand our discussion about policy evaluation into off-policy learning and nonlinear function approximation, where the states $\{S_t\}$ are drawn according to a behavior policy μ instead of the target policy π. True gradient algorithms like Gradient TD methods and RG remain convergent to local minima under off-policy training with any function approximator [Baird, 1995; Sutton *et al.*, 2009; Maei, 2011]. However, the empirical success of Gradient TD methods is limited to simple domains due to its large variance [Sutton *et al.*, 2016]. Semi-gradient algorithms are not convergent in general, e.g., the divergence of off-policy linear TD is well-documented [Tsitsiklis and Van Roy, 1997].

Semi-gradient algorithms are fast but in general not convergent. Residual gradient algorithms are convergent but slow [Baird, 1995]. To take advantage of both, [Baird, 1995] proposes to mix semi-gradients and residual gradients together, yielding the *residual algorithms*. The RA version of TD [Baird, 1995] updates $\mathbf{w}$ as

$$\begin{aligned}\mathbf{w} \leftarrow \mathbf{w} - \alpha\big(&R_{t+1} + \gamma \mathbf{v}(S_{t+1}) - \mathbf{v}(S_t)\big) \\ &\cdot\big(\gamma\eta \nabla_{\mathbf{w}} \mathbf{v}(S'_{t+1}) - \nabla_{\mathbf{w}} \mathbf{v}(S_t)\big),\end{aligned}$$

where $\eta \in [0, 1]$ controls how the two gradients are mixed. Little empirical study has been conducted for RA.

Control: We now consider the problem of control, where we are interested in finding an optimal policy π^* such that $q_{\pi^*}(s,a) \geq q_\pi(s,a)\, \forall(\pi, s, a)$. We use Q, parameterized by θ, to denote our estimate of q_π. Q-learning [Watkins and Dayan, 1992] is usually used to train Q and enjoys convergence guarantees in the tabular setting. When the action space is continuous, DDPG [Lillicrap *et al.*, 2015] is usually used as a continuous version of Q-learning. In DDPG, an actor $\mu : \mathcal{S} \to \mathcal{A}$, parameterized by ν, is trained to output the greedy action. DDPG updates θ and ν as

$$\begin{aligned}\theta \leftarrow \theta + \alpha_1\big(&r_{t+1} \\ &+ \gamma \bar{Q}(s_{t+1}, \bar{\mu}(s_{t+1})) - Q(s_t, a_t)\big)\nabla_\theta Q(s_t, a_t),\end{aligned} \tag{2}$$

$$\nu \leftarrow \nu + \alpha_2 \nabla_a Q(s_t, a)|_{a=\mu(s_t)} \nabla_\nu \mu(s_t), \tag{3}$$

where α_1, α_2 are step sizes, $\bar{\mu}, \bar{Q}$ are target networks [Mnih *et al.*, 2015; Zhang *et al.*, 2021], which are synchronized with μ, Q periodically.

DDPG is a semi-gradient algorithms. There are also true gradient methods for control, e.g., Greedy-GQ [Maei *et al.*, 2010] and the residual version of Q-learning [Baird, 1995]. As with Gradient TD methods, the empirical success of Greedy-GQ is limited to simple domains due to its large variance [Sutton *et al.*, 2016].

3 Residual Algorithms in Model-free RL

In this section, we investigate how to combine RA and DDPG. In particular, we consider almost deterministic environments where the double sampling issue is not significant.

In semi-gradient algorithms, value propagation goes backwards in time. The value of a state depends on the value of its successor through bootstrapping, and a target network is used to stabilize this bootstrapping. RA allows value propagation both forwards and backwards. The value of a state depends on the value of both its successor and predecessor. Therefore, we need to stabilize the bootstrapping in both directions. To this end, we propose the *bidirectional target network* technique. Employing this in DDPG yields Bi-Res-DDPG, which

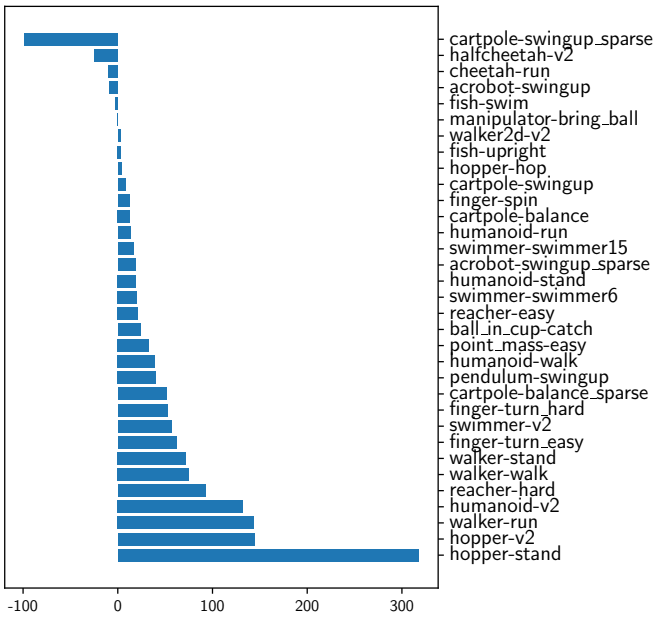

Figure 1: AUC improvements of Bi-Res-DDPG over DDPG on 28 DMControl tasks and 5 Mujoco tasks, computed as $\frac{\text{AUC}_{\text{Bi-Res-DDPG}} - \text{AUC}_{\text{DDPG}}}{\text{AUC}_{\text{DDPG}}}$.

updates the critic parameters θ as:

$$\begin{aligned}\theta \leftarrow \theta &- \alpha_1 \big(r_{t+1} + \gamma \bar{Q}(s_{t+1}, \bar{\mu}(s_{t+1})) - Q(s_t, a_t)\big) \\ &\times \big(-\nabla_\theta Q(s_t, a_t)\big) \\ &- \alpha_1 \big(r_{t+1} + \gamma Q(s_{t+1}, \mu(s_{t+1})) - \bar{Q}(s_t, a_t)\big) \\ &\times \eta\gamma \nabla_\theta Q(s_{t+1}, \mu(s_{t+1})),\end{aligned}$$

where $\bar{Q}, \bar{\mu}$ are target networks and $\eta \in [0, 1]$ controls how the two gradients are mixed. The actor update remains unchanged.

We compared Bi-Res-DDPG to DDPG in 28 DMControl tasks and 5 Mujoco tasks. Our DDPG implementation uses the same architecture and hyperparameters as [Lillicrap *et al.*, 2015], which are inherited by Bi-Res-DDPG (and all other DDPG variants in this paper). For Bi-Res-DDPG, we tune η over $\{0, 0.05, 0.1, 0.2, 0.4, 0.8, 1\}$ on `walker-stand` and use $\eta = 0.05$ across all tasks. We perform 20 deterministic evaluation episodes every 10^4 training steps and plot the averaged evaluation episode returns. In Figure 1, we report the improvement of AUC (area under the curve) of the evaluation curves. AUC serves as a proxy for learning speed (e.g., see Example 8.2 in [Sutton and Barto, 2018]). Bi-Res-DDPG achieves a 20% (41%) AUC improvement over the original DDPG in terms of the median (mean). Our DDPG baseline reaches the same performance level as the DDPG baseline in [Fujimoto *et al.*, 2018] and [Buckman *et al.*, 2018] in Mujoco tasks. An ablation study of Bi-Res-DDPG is provided in [Zhang *et al.*, 2020] to future investigate the necessity of the bidirectional target network.

We do not expect residual updates to improve the performance of all semi-gradient baselines. However, our results do show that the residual update together with the bidirectional target network is beneficial in many tasks. Despite the popularity of semi-gradient methods, we do believe residual algorithms deserve more study. The combination of residual updates and other semi-gradient algorithms, e.g., TD3 [Fujimoto *et al.*, 2018], is a possibility for future work. We also do not address the double sampling issue in stochastic environments. This is indeed a restriction, but we would like to emphasize that most available benchmarks with continuous actions have deterministic transitions, which indicates that this class of problems is of practical concern.

4 Residual Algorithms in Model-based RL

In model-based RL, the double sampling issue can be easily addressed by querying the learned model (either deterministic or stochastic). Given the empirical success of deterministic models and their robustness in complex tasks [Kurutach *et al.*, 2018; Feinberg *et al.*, 2018; Buckman *et al.*, 2018], we consider deterministic models in this paper. Dyna [Sutton, 1990] is a commonly used model-based RL framework that trains a value function with imaginary transitions from a learned model. In this paper, we consider the combination of Dyna and DDPG. For each planning step, we sample a transition (s, a, r, s') from a replay buffer and add some noise ϵ to the action a, yielding a new action $\hat{a}$. We then query a learned model with $(s, \hat{a})$ and get $(\hat{r}, \hat{s}')$. We aim to investigate different strategies for updating Q during planning.

One naive choice is to use the semi-gradient critic update (2). However, this suffers from the *distribution mismatch* problem [Feinberg *et al.*, 2018]. When we apply (2) in an imaginary transition $(s, \hat{a}, \hat{r}, \hat{s}')$, we need the Q-value on $\hat{s}'$ for bootstrapping. The Q-function is trained to make an accurate prediction on the state distribution of s, which is usually different from the state distribution of $\hat{s}'$. This distribution mismatch results from both an imperfect model and the different sampling strategies for a and $\hat{a}$. It yields an inaccurate prediction for $Q(\hat{s}', \mu(\hat{s}'))$, leading to poor performance [Feinberg *et al.*, 2018]. The TD(k) trick [Feinberg *et al.*, 2018] is one attempt to address this issue. With a real transition $(s_{-1}, a_{-1}, r_0, s_0)$ sampled from a replay buffer, a model is unrolled for k steps following $\bar{\mu}$, yielding a trajectory $(s_{-1}, a_{-1}, r_0, s_0, a_0, r_1, s_1, \ldots, r_k, s_k)$. TD($k$) then updates θ to minimize

$$\begin{aligned}&\tfrac{1}{k+1} \textstyle\sum_{t=-1}^{k-1} \Big(Q(s_t, a_t) \\ &- \big(\textstyle\sum_{i=t+1}^{k} \gamma^{i-t-1} r_i + \gamma^{k-t} \bar{Q}(s_k, \bar{\mu}(s_k))\big)\Big)^2. \qquad (4)\end{aligned}$$

With this update, Q is trained on distributions of almost all the states $(s_{-1}, \ldots, s_{k-1})$, which [Feinberg *et al.*, 2018] show helps performance. However, TD(k) still does not train Q on the last imaginary state s_k, which is used for bootstrapping. On the one hand, the influence of the bootstrapping error from s_k decreases as the trajectory gets longer thanks to discounting. On the other hand, even small state prediction errors typically compound as trajectories get longer, yielding a large prediction error of the state s_k itself. This contradiction is deeply embedded in TD(k). Consequently, TD(k)

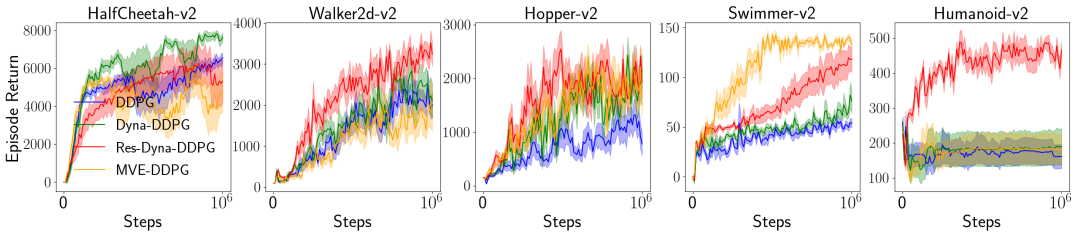

Figure 2: Evaluation performance for different model-based DDPG with an oracle model.

must assume the model is accurate for k-step unrolling, which is usually hard to satisfy in practice.

In this paper, we seek to mitigate this distribution mismatch issue through RA. For an imaginary transition $(s, \hat{a}, \hat{r}, \hat{s}')$, RA naturally allows the Q-function to be trained on both s and $\hat{s}'$, without requiring further unrolling like TD(k). The use of RA in model-based planning is inspired by the theoretical results from [Li, 2008], who proves that TD makes better predictions than RG. On a real transition, this accelerates backward value propagation by providing better bootstrapping. However, on an imaginary transition from a model, the value function is never trained on the imaginary successor state. It is questionable whether we should trust the value prediction on an imaginary state as much as a real state. We, therefore, propose to use RA on imaginary transitions, which encourages the Q-function to be consistent with the model as showed by [Li, 2008].

We now evaluate RA in model-based planning experimentally. We compare the performance of Dyna-DDPG (i.e., use the semi-gradient to update the critic with imaginary transitions), Res-Dyna-DDPG (i.e., use residual algorithms to update the critic with imaginary transitions), and MVE-DDPG (i.e., use the TD(k) trick to update the critic with imaginary transitions following [Feinberg *et al.*, 2018]). We consider five Mujoco tasks used by [Buckman *et al.*, 2018], which is a superset of tasks used by [Feinberg *et al.*, 2018]. In [Feinberg *et al.*, 2018], the unrolling steps of MVE-DDPG are different for different tasks, which serve as domain knowledge. For a fair comparison, [Buckman *et al.*, 2018] set $k = 3$ for all tasks in their baseline MVE-DDPG. In our empirical study, we followed this convention.

To separate planning from model learning, we consider planning with an oracle model. In this section, we restrict our empirical study on Mujoco tasks as we do not have direct access to the oracle models in DMControl tasks. We tune hyperparameters for Dyna-DDPG and Res-Dyna-DDPG on Walker and set $\eta = 0.2$ for all tasks. See [Zhang *et al.*, 2020] for more details. The results are reported in Figure 2. Curves are averaged over 8 independent runs and shadowed regions indicate standard errors. Both Dyna-DDPG and MVE-DDPG with an oracle model improve performance in 2 of 5 games, while Res-Dyna-DDPG improves performance in 4 out of 5 games. These results suggest that RA is a more effective approach to exploit a model for planning. In HalfCheetah, both MVE-DDPG and Res-Dyna-DDPG fail to outperform Dyna-DDPG. This could suggest that the distribution mismatch problem is not significant in this task. Furthermore, MVE-DDPG exhibits instability in HalfCheetah, which is also observed by [Buckman *et al.*, 2018]. The results of planning with a learned model is provided in [Zhang *et al.*, 2020], where the relative performance is similar to that of planning with an oracle model.

Note in Res-Dyna-DDPG, we use the naive residual algorithms without the bidirectional target network. Our preliminary experiments show that introducing the bidirectional target network during planning does not further boost performance. The main purpose of a target network is to stabilize bootstrapping (value propagation). Due to the distribution mismatch problem on imaginary transitions, however, it may be more important for the value function to be consistent with the model than simply propagating the value in either direction. This may reduce the importance of the bidirectional target network.

5 Conclusions

In this paper, we compare two classical RL methods, RG and TD, in both model-based and model-free settings. See [Zhang *et al.*, 2020] for a detailed review of related work. We propose the bidirectional target network technique to stabilize bootstrapping in both directions in RA, yielding a significant performance boost. We also demonstrate that RA is a more effective approach to the distribution mismatch problem in model-based planning than the existing TD(k) method. Our empirical study showed the efficacy of RA in deep RL problems, which has long been underestimated by the community. A possibility for future work is to study RA in model-free RL with stochastic environments, where the double sampling issue cannot be trivially resolved.

Acknowledgments

SZ is generously funded by the Engineering and Physical Sciences Research Council (EPSRC). This project has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713). The experiments were made possible by a generous equipment grant from NVIDIA.

References

[Baird, 1995] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. *Machine Learning*, 1995.

[Bellemare *et al.*, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013.

[Bellman, 1957] Richard E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[Buckman *et al.*, 2018] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, 2018.

[Feinberg *et al.*, 2018] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[Fujimoto *et al.*, 2018] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[Kurutach *et al.*, 2018] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

[Li, 2008] Lihong Li. A worst-case comparison between temporal difference and residual gradient with linear function approximation. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Maei *et al.*, 2010] Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[Maei, 2011] Hamid Reza Maei. *Gradient temporal-difference learning algorithms*. PhD thesis, University of Alberta, 2011.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[Puterman, 2014] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction (2nd Edition)*. MIT press, 2018.

[Sutton *et al.*, 2009] Richard S Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

[Sutton *et al.*, 2016] Richard S Sutton, A Rupam Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 2016.

[Sutton, 1988] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

[Sutton, 1990] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, 1990.

[Tassa *et al.*, 2018] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[Tsitsiklis and Van Roy, 1997] John N Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. In *Advances in Neural Information Pprocessing Systems*, 1997.

[Watkins and Dayan, 1992] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 1992.

[Zhang *et al.*, 2020] Shangtong Zhang, Wendelin Boehmer, and Shimon Whiteson. Deep residual reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, 2020.

[Zhang *et al.*, 2021] Shangtong Zhang, Hengshuai Yao, and Shimon Whiteson. Breaking the deadly triad with a target network. *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, 2021.