# Mixture of Step Returns in Bootstrapped DQN

**Po-Han Chiang** [*1] **Hsuan-Kung Yang** [*1] **Zhang-Wei Hong** [1 2] **Chun-Yi Lee** [1]

## Abstract

The concept of utilizing multi-step returns for updating value functions has long been adopted in the reinforcement learning domain. Conventional methods such as TD ($\lambda$) further extend this concept and use a single target value equivalent to an exponential average of different step returns. Nevertheless, different backup lengths provide diverse advantages in terms of bias and variance of value estimates, convergence speeds, and learning behaviors of the agent. Integrating step returns into a single target sacrifices the advantages offered by different step return targets. In order to address this issue, we propose Mixture Bootstrapped DQN (MB-DQN) and employ different backup lengths for different bootstrapped heads. MB-DQN enables diversity of the target values that is unavailable in approaches relying only on a single target value.

## 1. Introduction

In recent value-based deep reinforcement learning (DRL), a value function is usually utilized to evaluate state values, which stand for estimates of the expected long-term cumulative rewards that might be collected by an agent. In order to perform such an evaluation, a deep neural network (DNN) is employed by a number of contemporary value-based DRL methods (Mnih et al., 2015; Wang et al., 2016; Hasselt et al., 2016; Osband et al., 2016; Hessel et al., 2018) as the value function approximator, in which the network parameters are iteratively updated based on the agent's experience of interactions with an environment. For many of these methods (Mnih et al., 2015; Wang et al., 2016; Hasselt et al., 2016; Osband et al., 2016; Hessel et al., 2018), the update procedure is carried out by one-step temporal-difference (TD) learning (Sutton & Barto, 1998) (or simply "*one-step TD*"), which calculates the error between an estimated state value and a target differing by one timestep. One-step TD has been shown effective in backing up immediate reward signals collected by an agent. However, the long temporal horizon that the reward signals from farther states have to propagate through might lead to an extended learning period of the value function approximator.

Learning from multi-step returns (Sutton & Barto, 1998) is a way of propagating the rewards newly observed by an agent faster to earlier visited states, and has been adopted in several previous works. Asynchronous advantage actor-critic (A3C) (Mnih et al., 2016) employs multi-step returns as targets to update the value functions of its asynchronous threads. Rainbow deep Q-network (Rainbow DQN) (Hessel et al., 2018) also utilizes multi-step returns during the backup procedure. The authors in (Barth-Maron et al., 2018) also modify the target value function of deep deterministic dolicy gradient (DDPG) (Lillicrap et al., 2016) to estimate TD errors using multi-step returns. Updating value functions with different backup lengths provides advantages in different aspects in terms of bias and variance of value estimates, convergence speeds, and learning behaviors of the agent. Backing up reward signals through multi-step returns shifts the bias-variance tradeoff (Hessel et al., 2018). Therefore, backing up with different step return lengths (or simply '*backup length*' hereafter (Asis et al., 2018)) might lead to different target values in the Bellman equation, resulting in different learning behaviors of an agent as well as different achievable performance of it. The authors in (Amiranashvili et al., 2018) have demonstrated that the performance of an agent varies with different backup lengths, and showed that both very short and very long backup lengths could cause performance drops. These insights suggest that identifying the best backup length for an environment is not straightforward. In addition, although learning based on multi-step returns enhances the immediate sensitivity to future rewards, it is at the expense of greater variance which may cause the value function approximator to require more data samples to converge to the true expectation. Relying on a single target value with any specific backup length may also constrain the learning behavior and possible performance of the agent.

Based on the above observations, there have been several research works proposed to unify different target values

[*]Equal contribution [1]Elsa Lab, Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan [2]Department of Computer Science, Massachusetts Institute of Technology, USA. Correspondence to: Hsuan-Kung Yang <hellochick@gapp.nthu.edu.tw>, Chun-Yi Lee <cylee@cs.nthu.edu.tw>.

*Workshop on Reinforcement Learning Theory in International Conference on Machine Learning*, Copyright 2021 by the author(s).

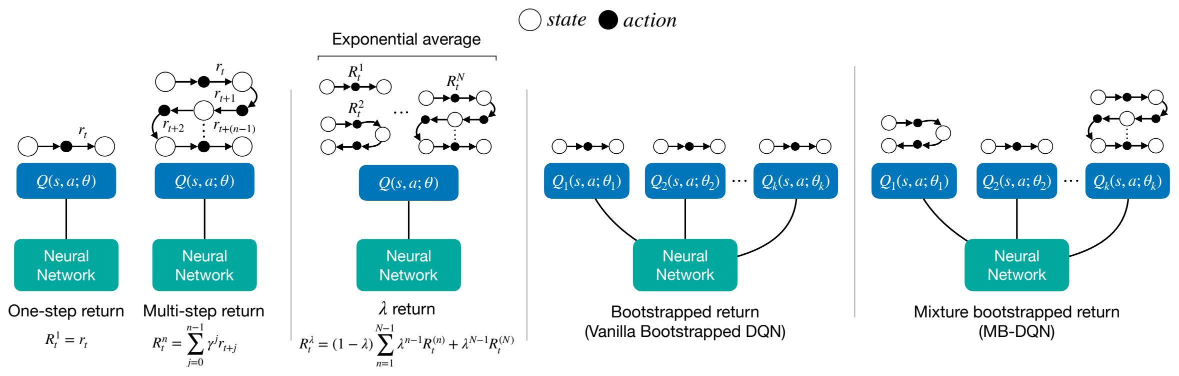

*Figure 1.* A summary of different return target schemes and the proposed MB-DQN framework.

with different backup lengths to leverage their respective advantages. The traditional TD ($\lambda$) (Sutton & Barto, 1998) uses a target value equivalent to an exponential average of all $n$-step returns (where $n$ is a natural number), offering a faster empirical convergence by interpolating between low-variance TD returns and low-bias Monte Carlo returns. DQN ($\lambda$) (Daley & Amato, 2019) further proposes an efficient implementation of TD ($\lambda$) for DRL by modifying the replay buffer such that $\lambda$-returns can be pre-computed. Although these methods benefit from combining multiple distinct backup lengths, they still rely on a single target value during the update procedure. Integrating step returns into a single target value, however, may sacrifice the diversity of the advantages from different step return targets.

As a result, in this paper, we propose **M**ixture **B**ootstrapped **DQN** (abbreviated as "*MB-DQN*") to address the above issues. MB-DQN is built on top of bootstrapped DQN (Osband et al., 2016), which contains multiple bootstrapped heads with randomly initialized weights to learn a set of value functions. MB-DQN leverages the advantages of different step return targets by assigning a distinct backup length to each bootstrapped head. Each bootstrapped head maintains its own target value derived from the assigned backup length during the update procedure. Since the backup lengths of the bootstrapped heads are distinct from each other, MB-DQN provides diversity in the target values as well as diversified learning behaviors of an agent that is unavailable in approaches relying only on a single target value. In summary, the primary contributions of the paper include the following: (1) introducing an approach for maintaining the advantages from different backup lengths, and (2) preserving diversity in the target values provided by different step return targets, and thus enhancing the boosting effect of the voting process from the boostrapped heads, and (3) investigating the performance gain offered by MB-DQN through an offline RL (Levine et al., 2020; Agarwal et al., 2020) perspective so as to ablatively and fairly justifying the advantages of MB-DQN.

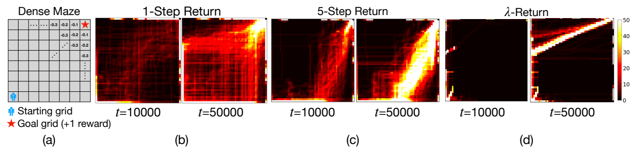

*Figure 2.* A visualization of the behaviors of the agents.

## 2. Methodology

### 2.1. Behavior Comparison of the DQN Agents with Different Designs of the Return Targets

To compare and highlight the impacts of different designs of the return targets on an agent's behavior, we employ an example maze environment containing a starting point and a goal, as depicted in Fig. 2 (a). In this environment, DQN is set as the default algorithm for the agents. We compare three different designs of the return targets: 1-step, 5-step, and $\lambda$-return (Daley & Amato, 2019), as illustrated in the first three architectures in Fig. 1, respectively. We depict the states visited by the corresponding agents within 50k timesteps during the training phase in Figs. 2 (b), (c), and (d), respectively. From the former two cases, it is observed that the agent trained with 5-step return reaches the goal through a clearer and more concentrated path than the agent trained with 1-step return. This is because the longer backup length allows the 5-step learner to adjust its value function estimation faster, thus guides it to quickly finds a path to follow and stablizes its policy. On the other hand, although the policy of the 1-step learner might converge slower than that of the 5-step learner, it is observed that 1-step return enables the agent to visit and explore more states in the early stage. This is because the reward signal from a farther state has to propagate through a longer temporal horizon. Therefore, the 1-step learner explores more extensively before learning an effective policy to reach the goal. Lastly, it can be observed that the agent trained with $\lambda$-return also quickly converges, and reaches the goal through a even more concentrated path. However, Fig. 2 (d) also reveals that the $\lambda$ learner explores apparently fewer states than those visited by the other two

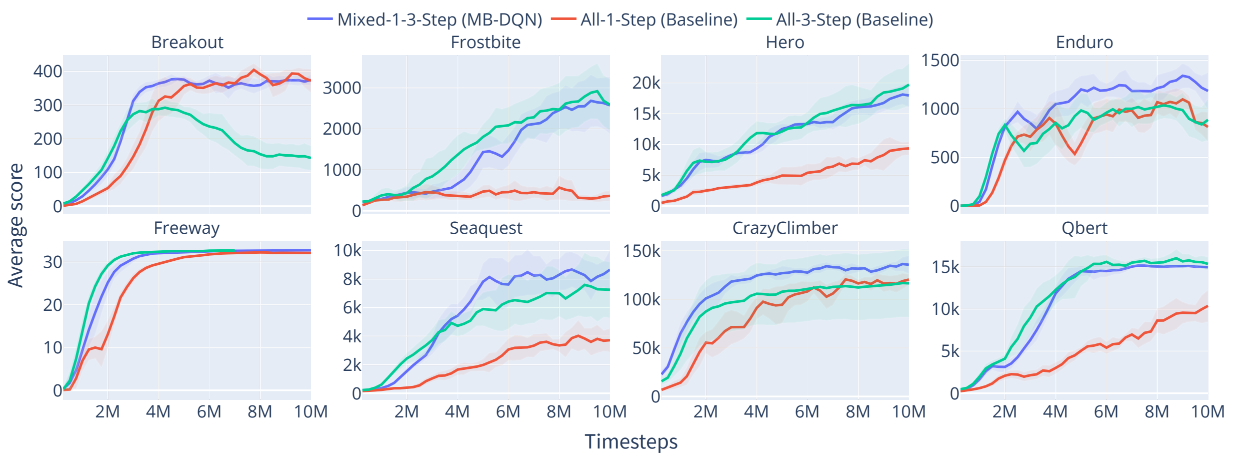

*Figure 3.* Comparison of the evaluation curves of MB-DQN and the baselines in eight *Atari* games. [2]

cases. The rationale behind this observation might be due to the even longer horizon of the backup lengths (up to 20) used by (Daley & Amato, 2019) for deriving the single $\lambda$-return target value. This motivational experiment demonstrates and validates that different designs of the return targets may lead to different behavior policies of the agents.

## 2.2. Mixture of Step Returns in Bootstrapped DQN

Although the $\lambda$-return target is able to combine different return targets into a single one, the example presented in the previous section shows that it may also sacrifice the exploratory benefits of the diverse behaviors of the agents trained from different return targets. We hypothesize that preserving such diverse behaviors may be advantageous and crucial to the agent. As a result, in order to preserve the properties and benefits from different backup lengths, we choose bootstrapped DQN (Osband et al., 2016) as our backbone framework. We leverage the advantages of distinct Q-value function heads in bootstrapped DQN, and propose the usage of mixtured backup lengths for different bootstrapped Q-value function heads in our MB-DQN framework, which is illustrated on the right-hand side of Fig. 1. MB-DQN similarly contains $K$ bootstrapped heads for estimating the Q-value functions, where each bootstrapped head $k \in K$ correspond to its own backup length $n_k$. In each episode, MB-DQN also uniformly and randomly selects a head $k \in \{1, ..., K\}$, and stores the state transition data collected by the agent using this head into a replay buffer. The replay buffer is played back periodically to update the parameters of all the bootstrapped Q-value function heads as well as the shared convolutional neural network. Each head is trained with its own target network $Q_k(s, a; \theta_k^-)$ and its own target value $y_{s,a}^k$ with the multi-step return. The multi-step returns with different backup lengths provide diversity in step returns for the $K$ bootstrapped estimates, preserving the properties and benefits from different backup lengths.

## 3. Experimental Results

### 3.1. Quantitative and Qualitative Comparisons

We compare MB-DQN against two baselines: bootstrapped DQN with (a) all 1-step return heads and (b) all 3-step return heads, denoted as *All-1-Step (Baseline)* and *All-3-Step (Baseline)*, respectively. We use ten bootstrapped heads for both MB-DQN and the baselines. MB-DQN (denoted as *Mixed-1-3-Step (MB-DQN)*) is implemented using five bootstrapped heads with 1-step backup length and another five bootstrapped heads with 3-step backup lengths. Fig. 3 presents the qualitative comparison. It is observed that longer backup lengths do not always guarantee better performances — each environment may have its own favor. The *All-3-Step* baseline outperforms the *All-1-Step* baseline in five out of eight games, while it performs comparably with the *All-1-Step* baseline in two games and suffers from a considerable performance drop in *Breakout*. In contrast, the proposed MB-DQN is able to deliver performances comparable to the better-performing baseline, and sometimes demonstrate even better performances than the baselines in terms of the scores and the convergence speed of the curves.

### 3.2. $\lambda$-Target v.s. Mixtured Bootstrapped Targets

In order to validate our assumption in Section 1 that integrating step returns into a single target value may sacrifice the diversity of the advantages provided by different step return targets, in this section, we compare these strategies of combining step returns in several *Atari* environments. For the unified return target strategy, we consider a recently proposed method called *DQN* ($\lambda$) (Daley & Amato, 2019), which implements TD ($\lambda$) by pre-computing $\lambda$-returns using an additional cache for its replay buffer memory. On the other hand, MB-DQN employs a strategy that leverages $K$ bootstrapped heads, where each head $k \in K$ has its own target value. For a fair comparison of the influence of the

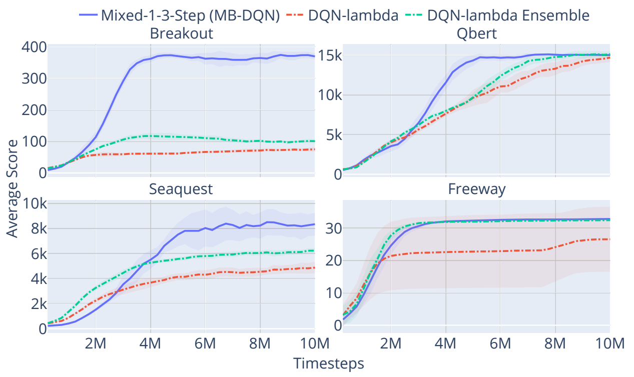

*Figure 4.* The evaluation curves for the comparison between the single $\lambda$-target strategy adopted by DQN ($\lambda$) and the multiple bootstrapped targets strategy adopted by MB-DQN.

bootstrapped heads, we further include a variant of DQN ($\lambda$), called *DQN ($\lambda$) Ensemble*, which employs $K$ bootstrapped heads using $\lambda$-return as the target value. In our experiments, We set $K = 10$ for MB-DQN and DQN ($\lambda$) Ensemble, where the settings for DQN ($\lambda$) are configured as its default values in (Daley & Amato, 2019). The single target value used by DQN ($\lambda$) is derived from multiple backup lengths ranging from one to a hundred. The evaluation curves of these strategies are plotted in Fig. 4. It can be observed that for the four environments presented in Fig. 4, the curves corresponding to the mixtured bootstrapped targets strategy (i.e., MB-DQN) grow faster and higher than those corresponding to the single-unified target strategy (i.e., DQN ($\lambda$) and DQN ($\lambda$) Ensemble). The above interesting evidence not only validates our assumption in Section 1, but also reveals that the advantages offered by the incorporation of multiple target values may outweigh the advantages offered by a single TD ($\lambda$) target that aggregates returns from the long temporal horizon.

### 3.3. Analysis of the Data Sample Quality and the Voting Stragety of MB-DQN

As the experimental results presented in the previous sections have quantitatively and qualitatively demonstrated the performance benefits offered by MB-DQN, we next dive further to investigate the rationale behind the advantages. We hypothesize that the performance improvements provided by MB-DQN may come from two possible causes: (1) the quality of the collected data samples in the experience replay buffer, and (2) the policy contributed from the voting of the bootstrapped heads with different backup lengths. In order to identify the main cause of the performance gain, we design another sets of experiment based on an offline RL setup (Levine et al., 2020; Agarwal et al., 2020). The use of the offline RL setups allows us to isolate an RL algorithm's ability to exploit experiences and generalize them from its ability to explore (Agarwal et al., 2020). Although the offline RL setup is oftern considered challenging due to the distribution mismatch between the current policy and the offline data collection policy (Levine et al., 2020; Agarwal et al., 2020), it is still an excellent choice for examining the rationale behind the performance offered by MB-DQN.

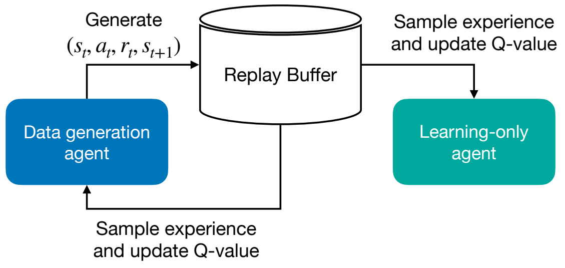
Generate
$(s_t, a_t, r_t, s_{t+1})$
Replay Buffer
Sample experience and update Q-value
Data generation agent
Learning-only agent
Sample experience and update Q-value

*Figure 5.* An overview of the offline learning setup.

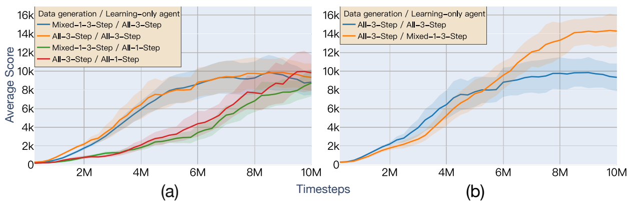

*Figure 6.* The evaluation curves for comparing different configurations of the *data generation agents* and the *learning-only agents* so as to investigate the performance benefits offered by MB-DQN.

The overview of the offline RL setup is depicted in Fig. 5, which consists two agents: one agent (denoted as the *data generation agent*) is responsible for generating state-action pairs for an experience replay buffer while updating its Q-value network with the data contained in it. The other agent (denoted as the *learning-only agent*) only updates its Q-value network by the existing data samples contained in the replay buffer, without contributing data to it (i.e., in an offline RL manner). In our experiments, both of these agents are implemented with ten bootstrapped heads. We consider two configurations for the *data generation agent*: *Mixed-1-3-Step (MB-DQN)* and *All-3-Step (Baseline)*, and two configurations for the *learning-only agent*: *All-1-Step (Baseline)* and *All-3-Step (Baseline)*. The experiments are evaluated on *Seaquest*, and the results corresponding to different configurations of the data generation agents and the learning-only agents are plotted in Fig. 6 (a), as indicated by the legends. It can be observed that even with different configurations of the data generation agents (i.e., either *Mixed-1-3-Step (MB-DQN)* or *All-3-Step (Baseline)*), the performances of the learning-only agents do not exhibit significant differences in terms of the scores and the learning speeds. The above observations thus suggest that the performance benefits of MB-DQN may not mainly come from the quality of the data samples collected in the relay buffer.

In order to validate the second hypothesis, we next introduce another experiment for comparing the performances of the offline learning-only agents based on the data samples collected by *All-3-Step (Baseline)*. The learning-only agents now include *All-3-Step (Baseline)* and *Mixed-1-3-Step (MB-DQN)* to inspect if the performance gain of MB-DQN results from the voting of the bootstrapped heads with different backup lengths. The results are presented in Fig. 6 (b), which reveals that with the same data samples contained in the experience replay buffer, the curve correspond to *Mixed-1-3-Step (MB-DQN)* is able to rise higher than that of the *All-3-Step (Baseline)* case. This observation indicates that the bootstrapped heads with different backup lengths do provide advantageous impact on the exploitation and generalization abilities of the agent. Since both bootstrapped DQN (i.e., using the same backup lengths for all of the heads) and MB-DQN (i.e., different heads may have different backup lengths) use multiple bootstrapped heads, the above evaluation results thus suggest that bootstrapped heads with different backup lengths may further enhance the boosting effect during the voting process.

## 4. Conclusion

We proposed MB-DQN for combining and leveraging the advantages of different step return targets using multiple bootstrapped heads. MB-DQN assigns a distinct backup length to each bootstrapped head, allowing it to enhance the boosting effect among boostrapped heads during the voting process, and preserve the diversity of the advantages from different step return targets. We first provided motivational examples, and then evaluated MB-DQN methodology on a number of *Atari 2600* environments. Moreover, we showed that incorporating multiple target values may outweight the advantaged offered by a single TD ($\lambda$) target. Finally, we inspected the benefits of MB-DQN through an offline RL setup, and showed that the voting of the bootstratpped heads plays a vital role in the performance of MB-DQN.

## References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. 2020.

Amiranashvili, A., Dosovitskiy, A., Koltun, V., and Brox, T. Analyzing the role of temporal differencing in deep reinforcement learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

Asis, K. D., Hernandez-Garcia, J. F., Holland, G. Z., and Sutton, R. S. Multi-step reinforcement learning: A unifying algorithm. In *Proceedings of AAAI Conference on Artificial Intelligence*, pp. 2902–2909, 2018.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. P. Distributed distributional deterministic policy gradients. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018.

Daley, B. and Amato, C. Reconciling $\lambda$ – returns with experience replay. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1133–1142. 2019.

Hasselt, H. v., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2094–2100, 2016.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3215–3222, 2018.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv:2005.01643*, 2020.

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. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, pp. 529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1928–1937, 2016.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4026–4034. 2016.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1995–2003, 2016.