Noname manuscript No.
(will be inserted by the editor)

Model-free Deep Reinforcement Learning – Algorithms and Applications

Reinforcement Learning Seminar – Winter Semester 2018/19

Fabian Otto

Received: date / Accepted: date

Abstract This survey presents an overview of the current model-free deep reinforcement learning landscape. It provides a comparison of state-of-the-art on-policy and off-policy algorithms in the value-based and policy-based domain. Influences and possible drawbacks of different algorithmic approaches are analyzed and associated with new improvements in order to overcome previous problems. Further, the survey shows application scenarios for difficult domains, including the game of Go, Starcraft II, Dota 2 and Rubik's Cube.

Keywords Deep Reinforcement Learning · Model-free · Neural Networks

1 Introduction

Artificial neural networks have been used in machine learning since the beginning. The first steps towards today's neural networks can be dated back to McCulloch and Pitts [26], who published the first work about the operating principles of neurons and created electrical circuits based on this idea. The following advancements of computers and Rosenblatt's invention of the perceptron [36] provided a boost to neural networks. Minsky and Papert followed up on this and stated in their book "Perceptrons" [27] that the perceptron algorithm is not able to learn non-linear behavior such as XOR problems. The consequent "AI Winter" was ended with the introduction of backpropagation [37], which provides an easy training procedure for multiple layers, therefore allows to model non-linear functions. Due to the lack of computational resources, the excitement decayed shortly after. More recently this bottleneck could be eliminated and neural networks have shown great success in e. g. computer vision [19,12], but have also found applications in reinforcement learning, such as the game of Go [43], Dota 2 [31] or StarCraft II [48]. Based on this success, this survey provides an overview of the current model-free reinforcement learning algorithm landscape for the on-policy and off-policy domain, additionally some notable application examples are presented.

Fabian Otto
Technische Universität Darmstadt, Computer Science Department
E-mail: fabian.otto@stud.tu-darmstadt.de

2 Background

A *Markov Decision Process* (MDP) is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathit{\Pi}(\mathcal{S})$ represents the state transition function, which returns a probability distribution over states. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ provides the expected immediate reward for taking each action in each state. This survey mainly considers the infinite horizon MDP, which tries to maximize the expected discounted cumulative reward:

$$G_t = \mathbb{E}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} \mathcal{R}(s_\tau, a_\tau) \middle| s_\tau, a_\tau\right]$$

Here $\gamma \in [0, 1)$ describes the discount factor, which trades off immediate and future rewards. Reinforcement learning algorithms now aim to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ which describes the optimal behavior maximizing the expected future reward from all states. Consequently, an action-value function which returns the value of taking an action a in state s, under the policy π is defined as:

$$Q_\pi(s, a) = \mathbb{E}\left[G_t | s_t = s, a_t = a, \pi\right] = \mathcal{R}(s, a) + \gamma \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s') \max_{a'} Q_\pi(s', a')$$

In order to find the best policy, value-based methods are interested in finding the optimal action-value function $Q^*(s, a) = \max_\pi Q_\pi(s, a)$, which can be computed iteratively by the Bellman operator:

$$Q_{t+1}(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s \in \mathcal{S}} \mathcal{T}(s, a, s') \max_{a'} Q_t(s', a')$$

When $Q_t \approx Q^*$, the optimal policy can be computed by $\pi^*(s) \approx \arg\max_a Q_t(s, a)$. Alternatively, the optimal value function $V^*(s) = \mathbb{E}\left[G_t | s_t = s, \pi\right] = \max_a Q^*(s, a)$ can be used.

In contrast to value-based methods, policy-based methods directly parameterize a policy $\pi(s; \theta)$. One such method is REINFORCE [52], it updates the policy parameters θ in the direction $\nabla_\theta \mathbb{E}[G_t] \approx \nabla_\theta \log \pi(s_t; \theta) G_t$. In order to reduce variance and keep the estimate unbiased, a baseline $b(s_t)$ is subtracted from the return $\nabla_\theta \log \pi(s; \theta)(G_t - b(s_t))$. For this baseline, it is common to use an estimate of the value function $b(s_t) \approx V_\pi(s_t)$. By applying the policy gradient theorem, $G_t - b(s_t)$ can be seen as an estimate of the advantage function $A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$. Combining parameterized policies and value function estimator results in *Actor-Critic* (AC) methods.

3 Algorithms

This section provides an overview of the current deep reinforcement learning landscape including improvements and state-of-the-art algorithms (Figure 3). This survey classifies algorithms either as on-policy or off-policy. On-policy algorithms estimate the value of the policy while using it for control. In contrast, off-policy algorithms utilize a generating policy, called *behavior policy*, and a *target policy*. The behavior policy is not necessarily related to the target policy, which is improved and evaluated. This allows to use a deterministic target policy, while sampling actions through a stochastic behavior policy. [46]

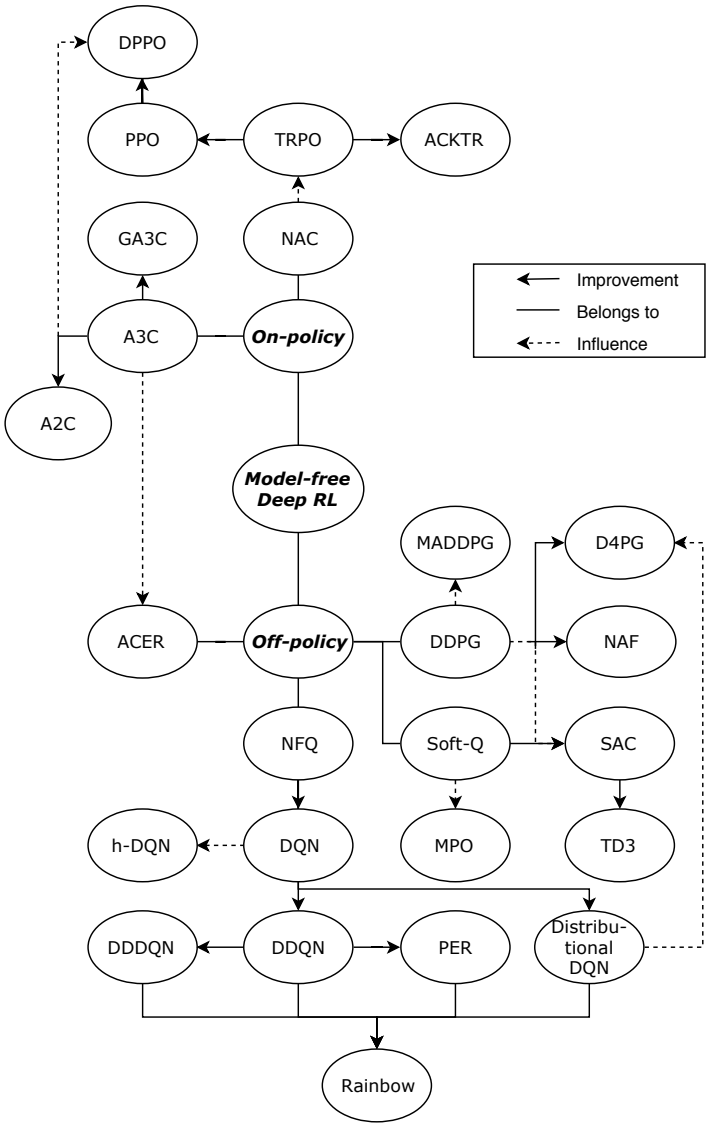

Fig. 1 Overview of model-free reinforcement learning algorithms.

3.1 Off-Policy

Many current off-policy algorithms are based on the original *Neural Fitted Q-Iteration* (NFQ) [34] published in 2005. NFQ is based on vanilla Q-Learning [51] and approximates the action-value function with a multilayer-perceptron. Compared to other approaches at the time, NFQ allows to learn with relatively good sample efficiency. Further, batches from a replay memory [22] are used in order to train the multilayer-perceptron with *Resilient Backpropagation* (RPROP) [35]. Thereby, it is possible to dynamically add new samples to the replay memory

during learning. Motivated by this idea as well as the success of TD-gammon [47] one of the biggest improvements in deep reinforcement learning, *Deep Q-Networks* (DQN) [29], was introduced. DQN shows human-level performance in multiple games of the Atari Arcade Environment [5]. Compared to NFQ, DQN introduces better scalability to large data sets by replacing RPROP with *Stochastic Gradient Descent* (SGD) updates. Further, DQN allows training end-to-end from the raw visual input utilizing *Convolutional Neural Networks* (CNN). However, the key improvement is the introduction of a second target Q-network, which is only periodically updated and thereby reduces sample correlations.
Despite the success, van Hasselt, et al. [11] show that DQN is not sufficient to avoid overestimation of action-values under certain conditions. This in itself is not harmful to the policy's performance, but if the overestimation is not uniform and not concentrated at the states of interest, it might affect the policy negatively. To mitigate the risk of overestimation, they propose *Double DQN* (DDQN). Double Q-Learning [10] in general decouples the selection and evaluation of actions by learning two action-value functions. One determines the greedy policy and the other the value of this policy. In order to reduce the computational cost, DDQN utilizes the already existing online Q-network for determining the greedy policy and the target Q-network for estimating its value. This allows to achieve better performance on most of the Atari games. Decoupling is also done in a more extreme form with *Dueling Double DQNs* (DDDQN) [50]. DDDQN represents the action-value by two separate function approximations, a state-value and a state-dependent action-advantage function estimate. This representation allows the dueling architecture to learn about the value of a state regardless of the action taken. This is especially important when the action has no repercussions on the near future. Combining DDQN with gradient clipping and *Prioritized Experience Replay* (PER) [38] shows that sampling rare experiences with a higher probability makes learning with replay memories even more efficient. However, sampling non-uniformly from the replay memory introduces a bias, that has to be corrected during the update step. The *51-atom distributional DQN agent* (C51) [4] does not compute the expected future reward, i. e. action-value function, but the corresponding distribution instead. This distributional view allows to reduce chattering and provides a more stable training as well as a rich set of auxiliary predictions. Combining different DQN versions in one algorithm shows even better performance and is introduced by *Rainbow* [14]. The authors additionally present an analysis of the incorporated components and show that all but one result in significantly improved performance.
All previous approaches are using environments, which do not have to deal with delayed and sparse feedback. In real world applications this is often not the case and algorithms still need the ability to learn. One key problem during training is insufficient exploration, causing unstable policies. Using intrinsically motivated agents, exploration can be achieved through the agent itself rather than through an external objective. This idea is implemented by *hierarchical-DQN* (h-DQN) [20]. A top level meta-controller is selecting a subgoal in order to optimize the extrinsic reward. The lower level controller maximizes an intrinsic reward by solving the subgoal. This enables h-DQN to achieve a significantly better performance on the Atari game "Montezuma's Revenge". In addition to that, real systems require to be able to work with continuous action spaces and need to ensure sample complexity is low, because of the more costly data collection process from e.g. robots.

Normalized Advantage Functions (NAF) [7] address both problems with a DQN motivated algorithm. Similar to DDDQN, the Q-network is represented by one state-value function and one state-dependent action-advantage function. In order to reduce sample complexity further, NAF incorporates model-based imagination rollouts. Consequently, the learned model is used to generate replay memory entries. As a drawback, NAF fails to show improvements using the model-based approach compared to on-policy samples, however it might be desirable when damage avoidance for real systems is crucial. In its core, NAF is mainly based on the more popular *Deep Deterministic Policy Gradients* (DDPG) [21] and tries reduce its complexity. DDPG does not follow an exclusively value-based learning and uses an AC approach with approximate Q-Learning in order to make deterministic policy gradients (DPG) [44] applicable for deep neural networks. The addition of batch normalization layers [16] enables to deal with different physical units in the observation space. For stabilizing policies, DDPG adapts the idea of target networks from DQN and uses soft target updates for AC. Following up on this, Haaranoja, et al. criticize that DDPG is brittle to hyper-parameter choices and the policy stability is difficult to achieve, as a result they present *Soft Actor Critic* (SAC) [9]. SAC is an improvement of *Soft-Q-Learning* (Soft-Q) [8], which combines the idea of Q-Learning with maximum entropy reinforcement learning. The soft action-value function allows the policy to represent complex multi-modal behavior and provides a natural exploration without adding additional noise. SAC avoids the complexity and potential instability of Soft-Q and forms a bridge between DDPG and stochastic policies by maximum entropy reinforcement learning. Similarly to NAF, they also approach the problem of sample efficiency and gain significant improvements compared to DDPG and several on-policy methods. In general, the authors found modeling the state-value function and the action-value function with two separate neural networks improved the stability of SAC. *Twin delayed DDPG* (TD3) [6] was developed in parallel to SAC and continues to address the hyper-parameter sensitivity of DDPG. Additionally, the authors show that overestimation of the action-value function is a pressing issue for AC methods as well. However, for the continuous action space the approach of DDQN is not easily applicable. Therefore, TD3 proposes three key improvements. A clipped variant of double Q-Learning trains two separate Q-Networks in order to reduce the overestimation bias. Policy smoothing adds noise to the target action, which avoids exploiting the action-value function errors and brings values of similar actions closer together. The most important improvement are delayed updates, i.e. the policy network as well as the target network are updated less frequently than the Q-network. This reduces the risk of optimizing the policy based on incorrect value estimates, hence the policy is less likely to diverge.

Building upon the idea of entropy reinforcement learning in Soft-Q, *Maximum a Posteriori Policy Optimization* (MPO) [1] transforms the reinforcement learning problem to an inference problem, which allows to utilize *Expectation Maximization* (EM) for training, while optimizing a relative-entropy objective. Similar to Rainbow, *Distributional Distributed DDPG* (D4PG) [3] combines different advancements with DDPG. This includes distributed actors, such as the on-policy *Asynchronous Advantage Actor Critic* (A3C) [28] and the distributional value estimate from C51. A3C also has an off-policy counterpart *Actor-Critic with Experience Replay* (ACER) [49], which outperforms A3C by a significant margin regarding performance and sample efficiency. This is achieved by Retrace [30] Q-value es-

timation, which reduces the bias of policy gradient estimates, whereas truncated importance weights help to reduce variance. Additionally, maintaining a running average of past policies forces the new policy to stay close to this average. This is similar to the *Kullback-Leibler* (KL) constraint in *Trust Region Policy Optimization* (TRPO) [39] but is computationally more efficient.
ACER uses multiple parallel agents to provide better exploration and faster convergence, however the goal was to find the optimal policy for one worker. *Multi-agent DDPG* (MADDPG) [23] on the other hand is interested in using multiple agents to collaborate and/or compete. The decentralized actors are only trained with local information, which allows for easier inference, the centralized critics have access to all information during training. In order to reduce the variance based on the interactions with other actors, policy ensembles are proposed.

3.2 On-Policy

Comparably important as DQN in the off-policy setting is the introduction of A3C [28] in the on-policy setting. Besides A3C, the authors introduce asynchronous off-policy methods but show that A3C outperforms all of them. Multiple actor threads allow A3C to collect more experience in a larger space of the environment and incorporate it into one globally shared network. Consequently, the stability and robustness of the training process is increased without a replay memory and less computational power is required to achieve better results compared to DQN. Similar to SAC, entropy is included in objective function, but A3C utilizes it for regularization instead of using a constraint. Aside from the discrete Atari games, A3C was also tested for continuous action spaces and was able to learn reliable policies. A2C [28] is a batched and synchronous variant of A3C and has shown better performance in practice by using only one worker with multiple environments [53], whereas GA3C [2] provides a framework to make A3C compatible with GPU computation.
Following the concept of policy gradients, Schulman, et al. introduce TRPO [39]. TRPO can be seen as combination of minorization-maximization [15] and *Natural Actor Critic* (NAC) [32]. They criticize that first order gradient methods are often overconfident and not accurate enough in curved areas, which results in losing learning progress already made. Therefore, TRPO optimizes its policy by enforcing a trust region constraint. The trust region constraint represents a lower bound on the performance of the policy and guarantees monotonic improvement locally around the current policy. The computational efficiency is improved by approximating the natural policy gradient [18] with the Conjugate Gradient algorithm. In order to compensate for the approximations, before applying the update, a backtracking line search determines whether the update still satisfies the constraint or not. One big drawback of TRPO in practice is its sample efficiency. Theoretically, TRPO can be applied to any policy, but it is often not practical to use deep neural network policies. This shortcoming is addressed by *Proximal Policy Optimization* (PPO) [40]. PPO keeps all monotonic improvement guarantees while using first order optimization. This can be achieved in two ways. The original constraint is replaced with regularization or alternatively, the importance sampling weights are clipped, which results in a lower bound for the unclipped objective. PPO can also be combined with A3C's idea of distributed actors resulting into *Distributed PPO*

(DPPO) [13]. Besides PPO, *Actor-Critic using Kronecker-factored Trust Region* (ACKTR) [53] approaches TRPO's shortcoming in sample complexity as well. ACKTR utilizes A3C's advantage function and approximates the natural gradient with *Kronecker-factored Approximate Curvature* (K-FAC) [24], which offers a computational cost comparable to SGD. By maintaining a running average of curvature information, K-FAC is able to reduce variance and achieves scalability, therefore ACKTR is more efficient in computing the inverse Fisher Information matrix. Further, ACKTR is not only able to increase the performance and sample efficiency compared to TRPO but also to A2C.

4 Applications

One of the earliest success stories of deep reinforcement learning is TD-gammon [47] achieving master level performance in backgammon. TD-gammon was starting with zero prior knowledge and improved exclusively by self-play. However, consecutive experiments aiming to reproduce the success in backgammon for other games such as chess, the game of Go and checkers were not successful. This was often attributed to the idea that state space exploration was helped by the stochasticity of the dice rolls [33]. Regardless of these failures, DQN was motivated by this success and showed that it was not only able to outperform existing methods on most of the Atari games, but even surpasses human-level performance. RL algorithms following after DQN were able to improve the performance even further (e.g. A3C) or specifically improve on poorly performing games, such as h-DQN on "Montezuma's Revenge".
Based on the idea of TD-gammon, AlphaGo [42] combines self-play with CNNs, supervised learning and *Monte Carlo Tree Search* (MCTS). In its core AlphaGo is selecting moves by a novel MCTS, which is guided by a learned value function and policy. However, instead of directly training from scratch, value function and policy are pretrained on human expert moves and are afterwards improved by self-play. In 2016 AlphaGo showed significantly better performance compared to other programs and was able to achieve the first victories over professional human players. AlphaGo Zero [45] eliminates the supervised learning aspect from AlphaGo and uses no human data or guidance beyond the basic rules of the game. Additional improvements include an updated version of MCTS and the use of MCTS during training and self-play. AlphaGo Zero was able to defeat the strongest human players without a single loss and also beat AlphaGo considerably. AlphaZero [43] generalizes AlphaGo Zero to the chess and shogi domain. As a consequence, AlphaZero has to take different outcomes, e.g. draws, into account. Further, it is not able to exploit symmetries such as in the game of Go. One major criticism about all three systems are the underlying computational requirements, e.g. training for AlphaZero was executed on 5,000 TPUs [17]. Therefore, it is questionable if the same algorithms perform equally well with less computational power.
Motivated by the idea of learning with zero prior knowledge in AlphaZero, DeepCube [25] applies this idea to Rubik's cube. DeepCube introduces *Autodidactic Iteration* (ADI), a policy iteration motivated algorithm for joint training of value and policy network. Small state space reductions were conducted in order to avoid redundancies. Similar to AlphaZero they combine replay memory and MCTS, however they do not only search for the winning move but also the shortest one. For

training DeepCube is working backwards and starts scrambling a solved cube an increasing amount of time to create more complex samples after longer periods of training.
Besides the Atari games other video games have been explored in the context of deep reinforcement learning. This includes Doom for which a recent approach [41] combines DDQN with a recurrent neural network and PER. Moreover, they utilize snapshot ensembles in order to get multiple networks and more robust predictions. One limitation is the selected game mode, which does not require movement of the character. This avoids trading off between movement and exploration of the virtual space and defeating enemies. Consequently, the network only needs to learn spatial awareness to focus closer targets first. The authors show improved results compared to previous methods, mainly based on the snapshot ensembling technique.
OpenAI's latest large scale project is OpenAI Five and OpenAI 1v1 bot for Dota 2 [31]. They use a highly scaled version of PPO incorporating an LSTM module approaching the complex problem space. The training is done without prior human knowledge, but similar to AlphaZero the required computation makes it hard to reproduce the results reliably. OpenAI Five is able to defeat human amateur teams, whereas OpenAI 1v1 even defeats professional human players. For Starcraft II, DeepMind's AlphaStar [48] shows that even in this challenging domain human-level performance is achievable. Similar to AlphaGo, AlphaStar combines supervised learning with reinforcement learning and incorporates an LSTM module such as OpenAI Five. Further, a league system is introduced, which builds upon population-based reinforcement learning and ensures exploration as well as avoids forgetting. This technique enables AlphaStar to defeat professional human players.

5 Conclusion and Discussion

Deep reinforcement learning has shown significant developments in the last years, including improvements for sample efficiency and performance on a variety of different tasks. Challenges such as the game of Go, Starcraft II or Dota 2 show nearly human level performance. In general, the literature indicates a trend towards off-policy learning methods, which allow to reuse samples via replay memories and consequently make these algorithms more sample efficient. Recently, maximum entropy-based reinforcement learning has become more popular and is combined with existing methods. Furthermore, algorithms like PPO find popularity due to their performance and simple implementation.
Albeit these results look promising, deep reinforcement learning is currently not at the point to be universally applicable to all problems and domains. Especially neural networks can show large performance differences depending on hyper-parameter choices. Empirical results show converging evidence that neural networks perform well in practice, but for now neural networks are not able to present any convergence guarantees. Based on this, algorithms like PPO often do not provide sufficient mathematical explanations supporting the practical success. Another issue neural networks encounter, is the catastrophic forgetting. Once the task is learned, the network might forget entirely how solve the task after continuing training. Consequently, future developments should try to address these issues in order to improve convergence and provided better interpretable results.

References

1. Abdolmaleki, A., Springenberg, J.T., Tassa, Y., et al.: Maximum a Posteriori Policy Optimisation. arXiv preprint arXiv:1806.06920 (2018)
2. Babaeizadeh, M., Frosio, I., Tyree, S., et al.: Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU (2017)
3. Barth-Maron, G., Hoffman, M.W., Budden, D., et al.: Distributed Distributional Deterministic Policy Gradients. In: International Conference on Learning Representations (ICLR) (2018)
4. Bellemare, M.G., Dabney, W., Munos, R.: A Distributional Perspective on Reinforcement Learning. In: International Conference for Machine Learning (ICML) (2017)
5. Bellemare, M.G., Naddaf, Y., Veness, J., et al.: The Arcade Learning Environment: An Evaluation Platform for General Agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013)
6. Fujimoto, S., van Hoof, H., Meger, D.: Addressing Function Approximation Error in Actor-Critic Methods. In: International Conference for Machine Learning (ICML), pp. 1582–1591 (2018)
7. Gu, S., Lillicrap, T., Sutskever, I., et al.: Continuous Deep Q-Learning with Model-based Acceleration. In: International Conference on Machine Learning (ICML), pp. 2829–2838 (2016)
8. Haarnoja, T., Tang, H., Abbeel, P., et al.: Reinforcement Learning with Deep Energy-Based Policies. In: International Conference on Machine Learning (ICML), pp. 1352–1361 (2017)
9. Haarnoja, T., Zhou, A., Abbeel, P., et al.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In: International Conference for Machine Learning (ICML) (2018)
10. van Hasselt, H.: Double Q-learning. In: Conference on Neural Information Processing Systems (NIPS), pp. 2613–2621 (2010)
11. van Hasselt, H., Guez, A., Silver, D.: Deep Reinforcement Learning with Double Q-learning. In: AAAI Conference on Artificial Intelligence, pp. 2094–2100 (2016)
12. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition (CVPR), pp. 770–778 (2016)
13. Heess, N., TB, D., Sriram, S., et al.: Emergence of Locomotion Behaviours in Rich Environments (2017)
14. Hessel, M., Modayil, J., Van Hasselt, H., et al.: Rainbow: Combining improvements in deep reinforcement learning. In: AAAI Conference on Artificial Intelligence (2018)
15. Hunter, D.R., Lange, K.: A tutorial on MM algorithms. The American Statistician **58**(1), 30–37 (2004)
16. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: International Conference on Machine Learning (ICML), pp. 448–456 (2015)
17. Jouppi, N.P., Young, C., Patil, N., et al.: In-Datacenter Performance Analysis of a Tensor Processing Unit. In: International Symposium on Computer Architecture (ISCA), pp. 1–12 (2017)
18. Kakade, S.: A Natural Policy Gradient. In: Conference on Neural Information Processing Systems (NIPS), pp. 1531–1538 (2001)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Conference on Neural Information Processing Systems (NIPS), pp. 1097–1105 (2012)
20. Kulkarni, T.D., Narasimhan, K., Saeedi, A., et al.: Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In: Conference on Neural Information Processing Systems (NIPS), pp. 3675–3683 (2016)
21. Lillicrap, T.P., Hunt, J.J., Pritzel, A., et al.: Continuous control with deep reinforcement learning. In: International Conference on Learning Representations (ICLR) (2016)
22. Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine Learning **8**(3), 293–321 (1992)
23. Lowe, R., WU, Y.I., Tamar, A., et al.: Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In: Conference on Neural Information Processing Systems (NIPS), pp. 6379–6390 (2017)
24. Martens, J., Grosse, R.: Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In: International conference on machine learning (ICML), pp. 2408–2417 (2015)

25. McAleer, S., Agostinelli, F., Shmakov, A., et al.: Solving the Rubik's Cube Without Human Knowledge. In: Conference on Neural Information Processing Systems (NIPS) (2018)
26. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics **5**(4), 115–133 (1943)
27. Minsky, M., Papert, S.A.: Perceptrons: An introduction to computational geometry. MIT press (2017)
28. Mnih, V., Badia, A.P., Mirza, M., et al.: Asynchronous methods for deep reinforcement learning. In: International conference on machine learning (ICML), pp. 1928–1937 (2016)
29. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
30. Munos, R., Stepleton, T., Harutyunyan, A., et al.: Safe and Efficient Off-Policy Reinforcement Learning. In: Conference on Neural Information Processing Systems (NIPS), pp. 1054–1062 (2016)
31. OpenAI: Openai five. https://blog.openai.com/openai-five/
32. Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing **71**(7-9), 1180–1190 (2008)
33. Pollack, J.B., Blair, A.D.: Why did TD-gammon work? In: Conference on Neural Information Processing Systems (NIPS), pp. 10–16 (1997)
34. Riedmiller, M.: Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In: European Conference on Machine Learning (ECML), pp. 317–328 (2005)
35. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: IEEE International Conference on Neural Networks, pp. 586–591 (1993)
36. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review pp. 65–386 (1958)
37. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
38. Schaul, T., Quan, J., Antonoglou, I., et al.: Prioritized Experience Replay. In: International Conference for Learning Representations (ICLR) (2015)
39. Schulman, J., Levine, S., Abbeel, P., et al.: Trust Region Policy Optimization. In: International Conference on Machine Learning (ICML), vol. 37, pp. 1889–1897 (2015)
40. Schulman, J., Wolski, F., Dhariwal, P., et al.: Proximal Policy Optimization Algorithms (2017)
41. Schulze, C., Schulze, M.: ViZDoom: DRQN with Prioritized Experience Replay, Double-Q Learning and Snapshot Ensembling. In: Intelligent Systems and Applications, pp. 1–17 (2018)
42. Silver, D., Huang, A., Maddison, C.J., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
43. Silver, D., Hubert, T., Schrittwieser, J., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
44. Silver, D., Lever, G., Heess, N., et al.: Deterministic Policy Gradient Algorithms. In: International Conference on Machine Learning (ICML), pp. 387–395 (2014)
45. Silver, D., Schrittwieser, J., Simonyan, K., et al.: Mastering the game of Go without human knowledge. Nature **550**(7676), 354–359 (2017)
46. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, second edition edn. MIT Press (2018)
47. Tesauro, G.: TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. Applications of Neural Networks **6**(2), 215–219 (1994)
48. Vinyals, O., Babuschkin, I., Chung, J., et al.: AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/ (2019)
49. Wang, Z., Bapst, V., Heess, N., et al.: Sample Efficient Actor-Critic with Experience Replay. In: International Conference on Learning Representations (ICLR) (2017)
50. Wang, Z., Schaul, T., Hessel, M., et al.: Dueling Network Architectures for Deep Reinforcement Learning. In: International Conference on Machine Learning (ICML), pp. 1995–2003 (2016)
51. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King's College, Cambridge (1989)
52. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8**(3), 229–256 (1992)
53. Wu, Y., Mansimov, E., Grosse, R.B., et al.: Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. In: Conference on Neural Information Processing Systems (NIPS), pp. 5279–5288 (2017)