A SURVEY ON VALUE-BASED DEEP REINFORCEMENT LEARNING

Feng Xiaolong
xlfeng886@163.com

.

ABSTRACT

Reinforcement learning (RL) is developed to address the problem of how to make a sequential decision. The goal of the RL algorithm is to maximize the total reward when the agent interaction with the environment. RL is very successful in many traditional fields for decades. From another aspect of AI technology, deep learning (DL) has become very popular in recent years and has extended the boundary of many tradition research filed. Combine with RL and DL is the fundamental idea for researchers recently. The first breakout comes from the DQN algorithm. We defined the new technology as deep reinforcement learning (DRL). DRL creates a new branch in both the RL field and the DL field. Since the DQN algorithm, there are many new algorithms that have been proposed, and the number of related papers has grown exponentially in recent years. In this paper, we focus on the value-based deep reinforcement learning method, which is one of the bases of the DRL algorithm sets. Many papers are derived from the original DQN algorithm.

1 INTRODUCTION

We can consider the essential question in computer science, how to build an efficient artificial intelligence (AI)? First, it must face the uncertain of the world and the complicated structure. Second, it should know how to make the sequence decision in the real world. Third, as (Sutton and Barto, 2018) discussed the mechanism of how humans affected by the dopamine, the AI agent should also be derived by the reward and towards the maximum of the total reward during the learning progress. In recent years, DRL is the most popular method to approach the efficient AI system. DRL utilizes the DL method to deal with the unstructured of the world. In the past decades, the researchers hand-designed the relevant features when they deal with the specific problem. The traditional pipeline cost too many human resources but can not increase the performance too much. With the rapid development of DL technology and its widespread application in various fields, state-of-art in many areas has been continuously refreshed. The reinforcement learning field is also one of the fields that deeply affected by DL technology. As we know, RL is inspired by the behavioral psychology (Sutton and Barto, 2018). It can handle the sequential decision-making problem. So, the combination of reinforcement learning and deep learning leads to a newborn deep reinforcement learning technology. The DRL technology fills us with visions of the future, and it has extended the boundaries of our cognition, allowing revolutionary changes in many applications.

We could find a lot of achievements brought by the DRL technology from (LeCun et al., 2015; Schmidhuber, 2015; Goodfellow et al., 2016). For example, (Mnih et al., 2015) utilized the DRL agent to learn the raw pixels of the Atari game and achieve human-level performance. (Silver et al., 2016) can defeat human champions in the go game. (Silver et al., 2018) use DRL to play chess and Shogi by the self-play method. (Vinyals et al., 2019) applied DRL in complicated strategy games and achieve a fantastic result. DRL also performs very well in real-world applications, such as robotics (Gandhi et al., 2017; Pinto et al., 2017), self-driving cars (Pan et al., 2017). We also can see some very successful application cases in the finance field (Deng et al., 2016).

Here we need to have an overview of the DRL algorithm taxonomy. As we can see, DRL is a fast-developing field, and it is not easy to draw the taxonomy accurately. We can have some traditional perspectives for this, which ignore the most advanced area of DRL (such as meta-learning, transfer learning, MAML exploration, and some other fields).

The root branching points in a DRL algorithm is to consider the question of whether we can access to the environment model. If the state transitions and rewards can be predicted from the simulated model, we can consider it as the model-based DRL algorithm. If we have no idea about the transitions and rewards, we should consider it as the model-free DRL algorithm.

For a model-free DRL algorithms, we could have A2C algorithm, A3C algorithm(Mnih et al., 2016), PPO algorithm (Schulman et al., 2017), TRPO algorithm (Schulman et al., 2015)), Q-learning algorithm (Mnih et al., 2013), C51 algorithm (Bellemare et al., 2017), QR-DQN algorithm (Dabney et al., 2018), ER algorithm (Andrychowicz et al., 2017)), TD3 algorithm (Fujimoto et al., 2018), SAC algorithm (Haarnoja et al., 2018), etc.

For a model-based DRL algorithm, it should consider whether its model is given or the model is unknown. Alpha Zero (Silver et al., 2018) is the application of the given model. If the model is unknown, we could have some algorithms such as I2A, MBMF, MBVE.

In this article, we only survey the value-based deep reinforcement learning.

2 PROBLEM FORMULATION

Reinforcement Learning (Sutton and Barto, 2018) is to learn how to make sequential decision when the AI agent interact with the environments $\mathbb{E}$. We will formalize the $\mathbb{E}$ as a Markov Decision Processes (MDPs). it could be described as the tuple (S, A, P, R). That means in each time step, the AI agent interact with the observed state $s_t \in S$, and chooses the action $a_t \in A$. The action will lead to the reward $r_t \sim R$ and next state $s_{t+1} \sim P$.

The purpose of the learning process is to maximize the total reward when the AI agent interacts with the environments $\mathbb{E}$. In the infinite case, we also consider the discount factor γ per time-step. The total discounted return at time step t should be $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T is the time-step and we ignore the step before time step t due to the causality law.

The optimal action-value function $Q^*(s, a)$ is the maximum expected return on the condition below s_t and a_t. We can define it as $Q^*(s, a) = \max_\pi \mathbb{E}[R_t|s_t = s, a_t = a, \pi]$, where π is the policy. The $Q^*(s, a)$ function will follow the Bellman equation and update the next Q value iteratively as $Q_{i+1}(s, a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s', a')|s, a\right]$. This is proved to converge to optimal Q^* in tabular case (Sutton and Barto, 2018). In the reinforcement learning community, this is typically a linear function approximator, but sometimes a non-linear function approximator is used instead, such as a neural network. When we use the neural network, we parameterize the Q-network according to the weight θ and train it by minimizing the mean square error (MSE) loss function. The parameter θ will change iteratively as below:

$$argmin_\theta \, L_i\left(\theta_i\right) = \mathbb{E}_{s,a\sim P(\cdot)}\Big(\mathbb{E}_{s'\sim E}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a] - Q\left(s, a; \theta_i\right)\Big)^2 \qquad (1)$$

In the optimization progress, previous parameters θ_{i-1} will be fixed. The optimization method could be one of the most popular methods, such as ADAM, RMSprop, Momentum SGD, etc.

3 RELATED WORK

For decades, scholars have made many outstanding contributions to the field of RL, and traditional RL algorithms are very mature. We could find many papers that review the traditional RL, such as (Gosavi, 2009; Kaelbling et al., 1996; Sutton and Barto, 2018; Szepesvári, 2010).

Benefit from the rapid development of DL technology, it is continuously refreshing the state-of-the-art in the RL field. There is also some survey cover the developments in the DRL field, such as (Arulkumaran et al., 2017). There is also some paper survey on the DRL application in NLP filed, such as (Ranzato et al., 2015) and (Bahdanau et al., 2016).

4 TRADITIONAL VALUE-BASED RL ALGORITHMS

The value-based RL algorithms will define the policy π, by constructing a function of the state value or state -action value. Q-learning algorithm (Watkins and Dayan, 1992) is the classical state-action value-based RL algorithm . We can find some traditional ways to improve the Q-learning by parameterized function approximator (Gordon, 1996).

Q-learning Algorithm: The simplest version of Q-learning is the tabular case, which looks up the Q value table during the training process. The Q-learning algorithm applies the Bellman equation to update the Q-value function (Bellman and Kalaba, 1962). The ieda is straightforward enough, but it works not very well in the high-dimensional state-action space. The parameterized value function $Q(s, a; \theta)$ must be used to fit the value. The detail of the basic Q-learning algorithm can refer to (Sutton and Barto, 2018).

Fitted Q-learning: As we could get a lot of experience, in the form of (s, a, r, s'). In fitted Q-learning (Gordon, 1996), it use an function approximator to update the $Q(s, a; \theta_k)$ in each iteration, where θ_k with respect to the parameter of the function in k_{th} iteration. So we could obtain the target value as the equation below:

$$y_k = r + \gamma \max_{a' \in A} Q(s', a'; \theta_k) \tag{2}$$

Neural Fitted Q-learning (NFQ): The neural fitted Q-learning (NFQ) (Riedmiller, 2005), The neural Q-network receives the state information as the input and output the probability of each action. The network is parameterized with a neural network $Q(s, a; \theta_k)$ and apply SGD to minimize the MSE. But the problem is that the target will be changed as the weights change and generate errors in the state-action space. We cannot guarantee the convergence by using this algorithm (Baird, 1995; Tsitsiklis and Van Roy, 1997; Gordon, 1996; Riedmiller, 2005). The other problem is the overestimate issue (Van Hasselt et al., 2016). The overestimate comes from the max operator when we estimate the value. The max operator will enlarge the noise during the training process.

5 FROM DQN TO RAINBOW

Here we specifically survey the deep Q-network (DQN) algorithm (Mnih et al., 2015). The DQN algorithm is the first algorithm that can achieve a superhuman level in the Arcade Learning Environment (ALE) (Bellemare et al., 2013). We will also survey various improvements that make up the Rainbow algorithm.

Deep Q-networks: There is some common sense as NFQ, and the DQN algorithm is so attractive due to the excellent performance in the variety of ATARI games. It directly learns the raw pixels from the video and keeps the same structure and parameters in all games. This is unimaginable in previous algorithm implementations, which face the curve of dimension problem in high dimension space. To address the unstable and divergence problem, (Mnih et al., 2015) have proposed two methods, experience replay, and target Q-network.

(a) Experience replay mechanism: DQN initializes a replay buffer $\mathbf{D}$ with transitions (s_t, a_t, r_t, s_{t+1}) and randomly choose samples from the buffer to train the deep neural network. The simple idea is very efficient during the implementation.

(b) Fixed target Q-network: To address the shift of the Q-value problem, the target Q-network will periodically synchronize parameters with the main Q-networks. The performance of DQN seems to be more stable for this technology.

Double DQN: To address the over-estimations issue, (Van Hasselt et al., 2016) proposes a solution to divide the max operation into two parts, action selection part and action evaluation part. Two networks evaluate and choose action values according to the loss function as below:

$$Loss = \Big[r_t + \gamma \hat{Q}\Big(s_{t+1}, \arg\max_{a_{t+1}} Q\big(s_{t+1}, a_{t+1}; \theta\big); \theta' \Big) - Q(s_t, a_t; \theta) \Big]^2 \tag{3}$$

The evaluation network is parameterized by θ', and apply $\epsilon - greedy$ policy. The evaluate network will periodically be synchronized with the estimate network.

Dueling DQN: We can decompose the Q-value function into value function and advantage function. So, the dueling DQN used a specialized dueling network architecture to implement this simple idea. (Wang et al., 2015) proposes to change the single output network to a double output network. They share the same encoder. The formulation can be considered as below:

$$Q(s,a;\theta_1,\theta_2) = V(s;\theta_1) + \Big(A(s,a;\theta_1) - \frac{\sum_{a'} A(s,a';\theta_1)}{|A|}\Big), \tag{4}$$

where θ_1 is the parameter of advantage function structure, and θ_1 is the parameter of the value function structure.

Prioritized Replay: In the DQN algorithm, it proposes to use replay buffer technology and the algorithm uniformly sample from the buffer to train the neural network. (Schaul et al., 2015) have a one more step ieda for this technology, the proposed prioritized replay buffer. They set the priority of the experience in the buffer according to the defined rules, and then sample the experience by the priority. The high priority experience is worse to learn. Priority rules are usually set using TD error, but the choice of this method could be defined according to the particular problem. So, it is not easy when we could not know what experience is essential and what is not.

Asynchronous Multi-step DQN: The previous DQN algorithm needs to consider using off-policy algorithms to improve efficiency, and it uses the replay buffer, which occupied much memory and computation resources. (Mnih et al., 2016) proposes an idea to use many agents to train the deep neural network. It applies the asynchronous gradient descent method to train the multiple agents. Each subprocess maintains its own environment, and all the accumulated gradient will be applied to the center. The n-step methods (Sutton and Barto, 2018) will be used to update the reward function, and it will be used to trade off the bias and variance problem.

Distributional DQN: Normally, the Bellman equation is used to calculate the expected reward value. But (Bellemare et al., 2017) introduces a different solution that uses the distributional perspective to update the Q-value function. Their motivation is to address the issue caused by the uncertainty of the system, which has multimodal distribution in action space.

In their work, they let $Z(s,a)$ be the return obtained by the current policy. Z is not a scalar but a distribution. Then we derived the distributional version of Bellman equation as follows: $Z(s,a) = r + \gamma Z(s',a')$.

This advanced proposal has been well used in practice (Bellemare et al., 2017), (Dabney et al., 2018), (Rowland et al., 2018). This method could lead to risk-aware behavior (Morimura et al., 2010) and leads to more robust learning in practice. The reason could be that the distributional perspective maybe provides more training signals than the previous method.

Noisy DQN: (Fortunato et al., 2017) proposal the Noisy Net. The noisy network could have the same structure as before, but its bias and weights could be perturbed iteratively. The motivations are to solve the limitations of exploring using ϵ-greedy policies. It usually will combine the deterministic and Gaussian noisy stream. In the implementation status, the randomized action-value function replaces the ϵ-greedy policy. Then, the MLP layers will be parameterized by the noisy network.

Rainbow DQN: (Matteo et al., 2017) proposes a solution that integrates all advantages of the seven methods, include DQN, double DQN, prioritized DDQN, dueling DDQN, multi-step, distributed DQN, and noisy DQN. Due to the ensemble of the seven methods, it called the Rainbow DQN. The combinations lead to a mighty learning algorithm, which is more stable, robust, and efficiency than any single method.

6 THE OTHER VALUE-BASED DRL ALOGRITHMS

Deep Q-learning from Demonstrations(DQfD): The motivation of the DQfD algorithm (Sendonaris and Dulac-Arnold, 2017) is to address the problem that a large amount of data is required during the training of the DQN algorithm. We have to face the fact that we cannot get enough

data from the real application. This issue also severely restricts the boundary of deep reinforcement learning. To take full advantage of the non-real-time data, which is generated from the demonstration of people or offline machines, The DQfD algorithm is proposed to accelerate the learning process by only a small amount of demonstration data. The core ieda of the DQfD algorithm is:

(a) Collect the demonstration data and add it to the replay buffer.

(b) Train the network with the demonstration data before the interaction process.

(c) It combines 1-step TD loss, L2 regularization loss, supervised loss, and n-step TD loss.

Sym DQN Algorithm: (Mahajan and Tulabandhula, 2017) proposes the Sym DQN algorithm to ensure the improvement of sampling efficiency through symmetries in complex DRL tasks. Symmetries in MDP can be defined by the concept of MDP homomorphisms. The definition can refer to the original article. Sym DQN contains two components, one is the similarity detecting procedure, and the other is the similarity incorporating. The previous component will calculate the similarity estimates the value, and the later component will help the learning of symmetric policies. The network training process can utilize the symmetries of the state-action pair and learn the state-action pair representation on the top of the network to output the same representation for the state-action pair.

Guided DQN Algorithm: (Taitler and Shimkin, 2017) proposes the Guided DQN algorithm to improve the performance of the original DQN algorithm. There is two basis improvement. The first ieda is to integrate with the prior knowledge during the learning process, for example, in the task of how to play StarCraft, the rules of the StarCraft game can guide the agent to choose the better action. The second improvement is to adjust the update interval time for the parameter of the target network. This improvement could avoid decreased performance when implementing the original DQN algorithm. Guided DQN chooses a short update interval at the beginning and chooses the larger update interval as the experience replay buffer increase. This method could make the reward more stable, and the selected policy could keep around the optimal policy.

LS-DQN Algorithm: (Levine et al., 2017) proposes the least squares deep Q-network(LS-DQN) algorithm to combine the DRL with the linear least square method. On the one hand, DRL can learn rich feature representations. On the other hand, using the least square method can make the learning process more stable. In their work, the last hidden layer should be repeatedly retrained base one the fitted Q iteration algorithm. To address the over-fitting issue, they use the bayesian regularization term for the least squares update. The experiment result shown the better performance of the LS-DQN algorithm than the DQN algorithm and DDQN algorithm in Atari games.

BBQN Algorithm: (Lipton et al., 2016b) proposes the Bayes-by-Backprop Q network to address the problem of worse performance when applying the ϵ-greedy exploration method in the high dimensional action space. In the process of action selection, the BBQN algorithm quantified the uncertainty of the Q function estimation and used it to guide the exploration process. In the DQN algorithm, the Q function will be represented by the neural network with parameter w. But in the BBQN algorithm, parameter w was considered as the random variable, and it will calculate the posterior probability distribution $q(w|\theta)$ of the parameter w. The posterior probability distribution q is the multivariable Gaussian distribution with diagonal covariance, that's mean the parameter w is sampled from the $N(\mu, \delta_i^2)$. The δ_i will be parameterized by $\delta_i = log(1 + exp(\rho_i))$ to make sure all the δ_i to be positive. Given the posterior probability distribution q, the Thompson sampling method is used for the exploration. The BBQN algorithm, which uses the Bayes-by-Backprop neural network to approximate the Q function, is shown to be faster than the traditional DQN algorithm.

DQN with penalty signal: (Leibfried et al., 2017) introduces the penalty signal for the DQN algorithm to address the overestimation issue. Their work is inspired by the concept of information theory, and the penalty signal could inject the randomness for reward function.

DOL Algorithm: (Mossalam et al., 2016) proposes Deep Optimistic Linear Support Learning (DOL) to handle the multi-objective decision issue. The DOL algorithm combines the optimistic linear support(OLS) (Roijers et al., 2015) framework with DQN. It obtains the vector w, which decides the importance between the multi-objective. Then, it minimize the inner product between the Q-value vector $Q(s, a; \theta)$ and w by optimize the parameter θ. To match the requirement of the OLS framework, they modify the output of the DQN architecture to As the learning process progresses, each goal will become more and more similar, that is, their corresponding optimal Q-values will become more and more similar. Because DQN can be used to extract the features, so the network parameters of the

previous target network can be reused to speed up the learning process of the next target network. According to how the network parameters are used, it is divided into two algorithms, Deep OLS Learning with Full Reuse(DOL-FR) algorithm and Deep OLS Learning with Partial Reuse(DOL-FR). The experiment result shows the DOL algorithm is much more accurately than the DQN algorithm.

Averaged-DQN Algorithm: (Anschel et al., 2017) proposes the averaged-DQN algorithm to address the instability and variability issue for the original DQN algorithm. This algorithm averages the previous Q value estimates to make the training process more stable, and improves performance by reducing target approximation error(TAE). The averaged Q value have the below formulation:

$$Q_t^{average} = \frac{1}{N}\sum_{n=1}^{N} Q(s, a; \theta_{t-n} \tag{5}$$

Averaged-DQN algorithm will replace the target as:

$$y_t = r + \gamma \max_{a' \in A} Q^{average}(s', a'; \theta_t) \tag{6}$$

This algorithm is a very semple extension for the original DQN algorithm, but the experiment has shown to be more stable than DQN algorithm.

DRQN Algorithm: (Hausknecht and Stone, 2015) proposes the Deep Recurrent Q-Network(DRQN) algorithm. This algorithm combines the LSTM network with DQN algorithm, and replace the MLP layers with the LSTM network. The combination could address the limited memory issue and the problem that the DQN relies on the complete game screen during the Atari task.

ADRQN Algorithm: (Zhu et al., 2017) proposes the Action-specific Deep Recurrent Q-Network (ADRQN) algorithm to improve the performance in partially observable Markov decision process(POMDP). They encode the history action and observation to feed the Q-network. ADRQN algorithm change the original transistion (s_t, a_t, r_t, s_{t+1}) to $(\{a_{t-1}, o_t\}, a_t, r_t, s_{t+1})$, and the improvement lead more efficiency in partially observable domains.

Bootstrapped DQN Algorithm: (Osband et al., 2016) proposes the bootstrapped DQN algorithm to address the low efficiency of the ϵ-greedy strategy. This algorithm combines deep exploration with DNN. It can effectively estimate the uncertainty of the DNN, and can achieve faster exponential speed learning than the convergence speed of the learning process of any random dithering strategy. This algorithm can be naturally extended to massively parallel systems.

DQN with log prior augmentation: (Jaques et al., 2017) proposes the Deep Q-learning with log prior augmentation algorithm to improve the structure and quality of sequence, which is generated by the RNN network. This algorithm also retains the information obtained from the data and the diversity of the samples. First, it will pre-train the LSTM network by maximum likelihood estimation. Second, it assigns initial weights to three neural networks, Q-network, target network, and reward RNN network. Among them, the connection weight matrix and bias of the reward RNN network remain unchanged during the training process, and the prior strategy $\pi(a_t|s_t)$ is output at the same time. To apply the RL method to the sequence generation task, the state at time t is defined as $s_t = \{a_1, a_2, \cdots, a_t\}$. The total reward is defined as the bellow equation with the constant c to trade off the importance of the reward.

$$r(s, a) = log\pi(a|s) + r_T(s, a)/c \tag{7}$$

Apply the reward to the DQN algorithm, they could get the target function as below:

$$L(\theta) = \mathbf{E}_\beta[(r(s, a) + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \tag{8}$$

Gorila DQN Algorithm: (Nair et al., 2015) apply the general reinforcement learning architecture to the DQN algorithm. The Gorila DQN has massively distributed architecture for DRL. The architecture

includes four components: parallel actors, parallel learners, distributed NN to represent the value, and the distributed storage of experience.

DARQN Algorithm: (Sorokin et al., 2015) proposes the deep attention recurrent Q-network (DARQN) algorithm to introduce the attention mechanism into DQN network, include the soft attention mechanism and the hard attention mechanism. The DARQN algorithm could select and focus on the ability to process smaller input image information regions, so it can also reduce the number of parameters in the deep neural network and the computational cost for training and testing.

DQN with Intrinsic Fear: (Lipton et al., 2016a) introduces intrinsic fear into the DQN to avoid the catastrophic states during the training. The key to the implementation is to train a danger model. It is a binary classification neural network, which has the same architecture with the DQN network except for the output layer. The danger model could predict the probability of the imminent catastrophe. The new algorithm changes the optimization target to the following equation.

$$y^{IntrinsicFear} = r + \max_{a'} Q(s', a'; \theta_Q) - \lambda \cdot d(s'; \theta_d) \tag{9}$$

The $(s'; \theta_d)$ represents the danger model, and λ is the fear factor, which trades off the importance of the intrinsic fear.

7 Conclusion

In the view of the theoretical significance and practical application value of DRL technology, this article surveys the value-based deep reinforcement learning technology. We mainly dived into the DQN algorithm and the improvement according to it. Many related improved algorithms are designed to solve a specific problem. According to the different emphasis on the improvement of the DQN algorithm, it can be divided into the improvement of training algorithm, improvement of neural network architecture, improvement of introducing new learning mechanism, and improvement based on newly proposed RL algorithm. We have only reviewed some of the improved algorithms, and there are still many other excellent algorithms for further work.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058.

Anschel, O., Baram, N., and Shimkin, N. (2017). Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 176–185. JMLR. org.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2016). An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Bellman, R. and Kalaba, R. (1962). Dynamic programming applied to control processes governed by general functional equations. *Proceedings of the National Academy of Sciences of the United States of America*, 48(10):1735.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2018). Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664.

Duryea, E., Ganger, M., and Hu, W. (2016). Exploring deep reinforcement learning with multi q-learning. *Intelligent Control and Automation*, 7(04):129.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. (2017). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.

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

Gandhi, D., Pinto, L., and Gupta, A. (2017). Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955. IEEE.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Gordon, G. J. (1996). Stable fitted reinforcement learning. In *Advances in neural information processing systems*, pages 1052–1058.

Gordon, G. J. (1999). Approximate solutions to markov decision processes.

Gosavi, A. (2009). Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838.

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

Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. (2017). Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1645–1654. JMLR. org.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Leibfried, F., Grau-Moya, J., and Bou-Ammar, H. (2017). An information-theoretic optimality principle for deep reinforcement learning. *arXiv preprint arXiv:1708.01867*.

Levine, N., Zahavy, T., Mankowitz, D. J., Tamar, A., and Mannor, S. (2017). Shallow updates for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3135–3145.

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

Lipton, Z. C., Azizzadenesheli, K., Kumar, A., Li, L., Gao, J., and Deng, L. (2016a). Combating reinforcement learning's sisyphean curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*.

Lipton, Z. C., Gao, J., Li, L., Li, X., Ahmed, F., and Deng, L. (2016b). Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*, 3.

Mahajan, A. and Tulabandhula, T. (2017). Symmetry learning for function approximation in reinforcement learning. *arXiv preprint arXiv:1706.02999*.

Matteo, H., Joseph, M., Hado, H., Tom, S., Georg, O., Will, D., Dan, H., Bilal, P., Mohammad, A., and David, S. (2017). Rainbow: Combining improvements in deep reinforcement learning. *arXiv: 1710.02298 v1 [cs. AI]*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010). Nonparametric return distribution approximation for reinforcement learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 799–806.

Mossalam, H., Assael, Y. M., Roijers, D. M., and Whiteson, S. (2016). Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., et al. (2015). Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034.

Pan, X., You, Y., Wang, Z., and Lu, C. (2017). Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*.

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. (2017). Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*.

Puterman, M. L. (2014). *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer.

Roijers, D. M., Whiteson, S., and Oliehoek, F. A. (2015). Computing convex coverage sets for faster multi-objective coordination. *Journal of Artificial Intelligence Research*, 52:399–443.

Rowland, M., Bellemare, M. G., Dabney, W., Munos, R., and Teh, Y. W. (2018). An analysis of categorical distributional reinforcement learning. *arXiv preprint arXiv:1802.08163*.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sendonaris, A. and Dulac-Arnold, C. G. (2017). Learning from demonstrations for real world reinforcement learning. *arXiv preprint arXiv:1704.03732*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., and Ignateva, A. (2015). Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103.

Taitler, A. and Shimkin, N. (2017). Learning control for air hockey striking using deep reinforcement learning. In *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, pages 22–27. IEEE.

Tsitsiklis, J. N. and Van Roy, B. (1997). Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pages 1075–1081.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., et al. (2019). Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind Blog*.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Zhu, P., Li, X., Poupart, P., and Miao, G. (2017). On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*.