Journal on Artificial Intelligence JAI, vol.2, no.1, pp.39-58, 2020

Survey on the Application of Deep Reinforcement Learning in Image Processing

Wei Fang[1, 2, 3, *], Lin Pang[1] and Weinan Yi[1]

Abstract: In recent years, with the rapid development of human society, more and more complex tasks have emerged that require deep learning to automatically extract abstract feature representations from a large amount of data, and use reinforcement learning to learn the best strategy to complete the task. Through the combination of deep learning and reinforcement learning, end-to-end input and output can be achieved, and substantial breakthroughs have been made in many planning and decision-making systems with infinite states, such as games, in particular, AlphaGo, robotics, natural language processing, dialogue systems, machine translation, and computer vision. In this paper we have summarized the main techniques of deep reinforcement learning and its applications in image processing.

Keywords: Reinforcement learning, image processing.

1 Introduction

Reinforcement learning problems is that learning what to do and how to map situation to actions, so as to maximize expected cumulative reward signals. The agents must discover how to interact with a dynamic, initially unknown environment, learning an optimal policy, by trial and error [Sutton and Barto (2018)]. With recent exciting achievements of deep learning [LeCun, Bengio and Hinton (2015); Goodfellow, Bengio and Courville (2016)], benefiting from powerful computation, new algorithmic techniques, mature software packages and architectures, RL can be better combined with deep learning. With the use of deep learning algorithms within reinforcement learning, deep reinforcement learning arises at the historic moment.

Although the application of general reinforcement learning is relatively successful, the feature state needs to be set manually, which is very difficult for complex scenes, especially prone to cause dimensional disaster. Because the feature of deep learning is automatic learning, deep learning is used to extract features and then applied to reinforcement learning. Both of them draw on each other's strengths and get good results.

[1] School of Computer & Software, Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, China.

[2] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China.

[3] Jiangsu Key Laboratory of Computer Information Process Technology, Soochow University, Suzhou, China.

[*] Corresponding Author: Wei Fang. Email: Fangwei@nuist.edu.cn.

Received: 27 January 2020; Accepted: 01 April 2020.

Deep reinforcement learning combines the perception ability of deep learning with the decision-making ability of reinforcement learning, which is an artificial intelligence method closer to human thinking mode [Kaelbling, Littman and Moore (1996)]. Under the operation of Google's DeepMind team, deep reinforcement learning has suddenly become very popular. This is attributed to the paper published by the DeepMind team in Nature [Mnih, Kavukcuoglu, Silver et al. (2015)], which published how the AI that is plays more games than people made. The application of deep reinforcement learning in games is to take the picture of the game interface as the input of policy and output the next strategy of the game. This is equivalent to an application in image processing. In recent years, deep reinforcement learning has been widely used in image processing. In this paper, we will summarize the concepts and methods used in those papers about image processing using deep reinforcement learning.

The outline of this survey is as follows. Firstly, we briefly introduce deep reinforcement learning in Section 2. Then, in Section 3, we have made a partial investigation on the application of deep reinforcement learning in image processing. Finally, we summarize the whole paper in Section 4.

2 Related work of deep reinforcement learning

In this section, firstly, we will briefly introduce deep learning and reinforcement learning. For deep learning, we do not give detailed background introduction and for reinforcement learning, we will introduce the basic conceptual principles, reinforcement learning methods, and algorithm such as Q-learning and Sarsa. Next, we will present the three basic algorithms DQN, policy gradient and PPO and the comparative introduction of various algorithms.

2.1 Deep learning brief summary and development

Deep learning is a complex machine learning algorithm, which essence is to construct a deep neural network model to fit complex probability distributions, thereby improving the accuracy of classification and prediction of samples. Hinton proposed a method to improve model training process in 2006 promoted the explosive development of deep learning [Hinton, Osindero and Teh (2006)]. In 2012, in the famous ImageNet image recognition competition, the AlexNet model achieved remarkable results and proved the ability of deep learning to the world. Subsequently, various deep learning models have emerged and are widely used in various fields, such as computer vision [Fang, Zhang, Sheng et al. (2018); Fang, Zhang, Ding et al. (2020)], natural language processing, speech recognition, planning decision systems, recommendation and personalization technologies, and other related fields, and have achieved state-of-the-art levels.

2.2 Basic concepts of reinforcement learning

The Reinforcement learning (RL) approach focuses on learning problem-solving strategies. RL has three characteristics: Firstly, RL is a closed loop problem. The action of learning system will also affect its subsequent input. Secondly, learner does not tell which action to take directly, but finds which action will produce the maximum reward value. Thirdly, the consequences of an action will not appear immediately [Sutton and Barto (2018)].

In the Section 2.2.1, we will introduce the elements of reinforcement learning. Then in

Section 2.2.2, we are going to analyze dynamic programming, which is a set of algorithms used to calculate the optimal strategy for a class of Markov decision processes (MDP). Finally, we will mention the reinforcement learning algorithm in Section 2.2.4.

2.2.1 Elements of reinforcement learning

a. policy

Policy is a function whose input is a state of environment and whose output is the action to be executed under the state. And it is the core of reinforcement learning agent, because a policy can independently determine the agent's behavior.

b. reward

Reward signal defines the goal of reinforcement learning problem, which represents whether an action is good or bad for the agent. Environment will send a scalar, a reward, to the agent for each time step. The only goal of agent is to maximize the total reward received over a long period of time.

c. value function

Value function means what is good in the long run, while reward signal means what is good in the immediate sense. The value of state is the total reward that the agent expects to accumulate from the current state in the future. Our goal is to find the action that can get the highest value, so that these actions can get the most reward in the long run, instead of settling for the biggest reward in the short run. The reward is directly given by environment, but the value needs to be observed in the whole execution time of the agent, so as to estimate and re-estimate the value.

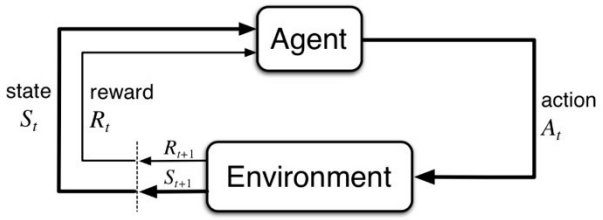

Figure 1: Adapted from Sutton et al. [Sutton and Barto (2018)]

Agent refers to learners and decision-makers, and the interaction with agent is called environment. This interaction is continuous as shown in Fig. 1. The agent chooses actions, environment responds to them and generates a new state. Meanwhile, environment generates a reward, which is the amount the agent wants to maximize with time.

2.2.2 Dynamic programming

a. Value function

Value functions evaluate the future rewards an agent expects to receive when it takes an action in a state. It is determined by a specific policy $\pi(a|s)$, which is the probability of taking action a in state s. The value of a state s under a policy π, denoted $v_\pi(s)$:

$$v_\pi = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad (1)$$

b. Optimal value functions

Policy is a parameterized function whose input is state and output is action. Since the task of reinforcement learning is to maximize the long-term reward, we need to find the optimal value function v_*, so that the execution of action a in state s can obtain the maximum value in the future. The policy corresponding to optimal value functions is called optimal policy π_*. Optimal value function is defined as follows:

$$v_*(s) = \max_{\pi} v_\pi(s) \quad \forall s \in S, v_*(s) \geq v_\pi(s) \tag{2}$$

2.3 Two basic reinforcement learning algorithm

2.3.1 Q-learning

Q-learning [Watkins (1989)] is a form of model-free reinforcement learning. Learning proceeds similarly to Sutton's [Sutton and Barto (2018)] method of temporal differences (TD): an agent tries an action at a particular state, and evaluates its consequences in terms of the immediate reward or penalty it receives and its estimate of the value of the state to which it is taken [Sutton (1988)]. The agent tries all actions under state s_1, and determines the best action by evaluating the reward received and the expectation of the future value, and then selects the best action to enter the next state s_2, repeating the process until the final state. The reward of the action under each state is stored in the Q table, and q-learning is the update of the Q table.

The purpose of agent is to maximize total discounted expected reward. Under a policy π, agent can choose action a_n at state s_n and receives a reward r_n, whose mean value $\mathcal{R}_{s_n}(a_n)$ only depends on the state and action the agent choose. The value of state s_n is as follow:

$$V^\pi(s) = \mathcal{R}_s(\pi(s)) + \gamma \sum_{s'} P_{ss'}(s'|s, \pi(s)) V^\pi(s') \tag{3}$$

This is a Bellman process. The agent select state s' that is maximize $V^\pi(s)$ with probability $P_{ss'}(s'|s, \pi(s))$ which is the distribution the state changes. According to the introduction of Section 2.2.2, we know that there is at least one optimal policy π^* which determine the optimal value function:

$$V^*(x) = V^{\pi^*}(x) = \max_a \{\mathcal{R}_s(a) + \gamma \sum_{s'} P_{ss'}(s'|s, a) V^{\pi^*}(s')\} \tag{4}$$

The task of Q-learner is to determining a π^* without initially knowing these values. We know that Q-learning is an incremental dynamic programming problem, which is determining the optimal policy step-by-step [Watkins (1989)]. For a policy π, define Q values as:

$$Q^\pi(s, a) = \mathcal{R}_s(a) + \gamma \sum_{s'} P_{ss'}(s'|s, \pi(s)) V^\pi(s') \tag{5}$$

Watkins et al. [Watkins and Dayan (1992)] define the Q values for an optimal policy as $V^*(x) = \max_a Q^{\pi^*}(s, a), \forall s, a$. The update of Q-learning can be expressed by the following equation:

$$Q_n(s, a) = \begin{cases} (1 - \alpha_n) Q_{n-1}(s, a) + \alpha_n [r_n + \gamma V_{n-1}(s'_n)] & if\ s = s_n\ and\ a = a_n, \\ Q_{n-1}(s, a) & otherwise \end{cases} \tag{6}$$

2.3.2 Sarsa

Sarsa, like Q-learning, makes decisions in the form of Q tables. Select the action with a large value from Q-table and apply it to the environment in exchange for rewards and punishments. The difference is that the update method of Sarsa is different. In Q-learning, it will first judge which action will bring the maximum reward $V^*(x)$ under state s, while the agent will not necessarily choose the action that brings the maximum reward, but only estimate the value of the following action. And the action that Sarsa estimates in state s is the action that is going to be done next. The algorithm of Sarsa is shown as follows.

Algorithm 1. The update of Sarsa

Initialize $\boldsymbol{Q(s,a)}$ arbitrarily
  for each episode **do**:
    Initialize $\boldsymbol{s}$
    for each step of episode **do**:
      Choose $\boldsymbol{a}$ from $\boldsymbol{s}$ using policy derived from Q (e.g., $\boldsymbol{\varepsilon - greedy}$)
      Take action $\boldsymbol{a}$, observe $\boldsymbol{r, s'}$
      Choose $\boldsymbol{a'}$ from $\boldsymbol{s'}$ using policy derived from Q (e.g., $\boldsymbol{\varepsilon - greedy}$)
        $\boldsymbol{Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]}$
        $\boldsymbol{s \leftarrow s'}$
      Until $\boldsymbol{s}$ is terminal
    end for
  end for

2.4 Introduction to deep reinforcement learning and algorithms

Traditional reinforcement learning has a problem. We use tables to store every state and in this state the Q value of each action has. But nowadays question is too complicated, the state is too numerous to enumerate (such as AlphaGo). If store all of them with the Q-table, it will not only cost a lot of computer memory, but also consume time to search the corresponding state in such a big table. However, in machine learning, neural network can be trained to predict output of the unknown input. We can use the state and action as the input of neural network, then the Q value of the action can be obtained after the analysis of the neural network. In another form, we can only input the state value, output all the action values, and then select the action with the maximum value as the next action according to the principle of Q learning. There are three basic deep reinforcement learning algorithms DQN, policy gradient and PPO, and a comparative introduction to various algorithms.

2.4.1 Introduction to deep Q network (DQN)

DQN is a combination of convolutional neural network and Q-learning [Mnih, Kavukcuoglu, Silver et al. (2013); Mnih, Kavukcuoglu, Silver et al. (2015)]. First, we use $\mathcal{R}_s + \gamma maxQ(s')$ to represent the correct Q value of action $a, \forall a \in a_n$, and $Q(s)$ to represent the estimated Q value, so as to realize the update of the neural network as follows:

$$\theta' \leftarrow \theta + \alpha\left(\left(\mathcal{R}_s + \gamma maxQ(s')\right) - Q(s)\right) \tag{7}$$

Neural network predicts the estimated value of Q-value after all actions are taken, and then we select the action with the maximum value in the estimation of Q-value to get rewards

in the environment. Then updates the parameters in the neural network use the above algorithm. There are two main factors that make DQN incredibly powerful, Experience replay and Fixed q-targets. The basic working principle is shown in Fig. 2.

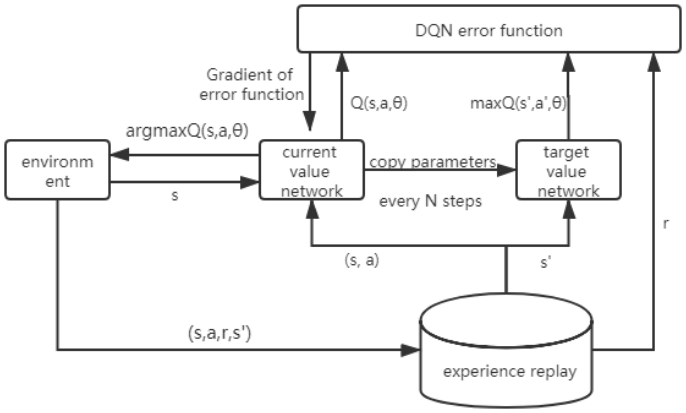

Figure 2: Deep Q-Network

In simple terms, DQN has a memory for learning experience before. Q learning is a kind of off - policy learning method, it can learn the experience, also can learn the past experience, and even learning the experience of others. When every step DQN update, we can randomly select some previous experience to learn. The method of experience replay disrupts the correlation between the experience, also makes it is more efficient to update neural network. Fixed Q-targets is also a kind of mechanism disturbing correlation. If use the fixed Q - targets, we will use two same structure but different parameters of the neural network in DQN. The estimate of Q-value was predicted using a neural network with the latest parameters, while the neural network predicting the actual value of Q-value used previously parameters.

2.4.2 Introduction to policy gradient

It is not realistic to calculate the value of an infinite number of actions by DQN. Policy Gradients uses a neural network Policy network to output predicted actions [Williams (1992)]. The biggest benefit of the Policy Gradients direct output of actions is that it can select actions within a continuum. However, policy gradient is a policy-based method, so policy network can also output probability.

The trajectory $\tau = \{s_1, a_1, s_2, a_2, \ldots, s_T, a_T\}$ express all the state agent experienced and action agent selected in an episode which has T step. Using θ represent parameters in policy network and $\pi_\theta(\tau)$ represent the probability of agent experience trajectory τ.

$$\pi_\theta(\tau) = \pi(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t)\pi(s_{t+1}|s_t, a_t) \tag{8}$$

The purpose of the neural network is to maximize the expected reward as follows:

$$\bar{R}_\theta = \sum_\tau R(\tau)\, \pi_\theta(\tau) = E_{\tau\sim\pi_\theta(\tau)}[R(\tau)] = E_{\tau\sim\pi_\theta(\tau)}[R(\tau)] \tag{9}$$

Gradient ascent algorithm is used to update the parameters of the neural network. Increase the probability $\pi_\theta(a_t^n|s_t^n)$ of an action with more rewards in step t of trajectory n and

decrease the probability of an action with fewer rewards.

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau)\, \pi_\theta(\tau) \nabla log \pi_\theta(\tau) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log \pi_\theta(a_t^n | s_t^n) \tag{10}$$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta \tag{11}$$

Ideally, if all the rewards are positive, all the π_θ will increase. But since the sum of probabilities is 1, the ones that increases less go down, and the ones that increases more go up. But in practice we are sampled from countless τ N times, if the action a has not been sampled, then the probability of selecting a will drop, but does not mean that the action a is bad. To solve this problem, adopt the method of add a baseline so that all rewards are not always positive. The gradient of $\bar{R}_\theta$ can be approximated as follows:

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (R(\tau^n) - b) \nabla log \pi_\theta(a_t^n | s_t^n) \tag{12}$$

In an episode, the execution of an action a_t in state s_t is always multiplied by the same weight $\frac{1}{N} \sum_{n=1}^{N} (R(\tau^n) - b)$, which is actually unfair, because some actions in an episode may be good and some may be bad. Even if the episode turns out to be good, it doesn't mean that all the actions in it are good. It is much better to multiply each action by a different weight, which reflects whether the action is good or not. Furthermore, we will give a discount reward in the future, because the action at a certain place will have less effect on the reward of the later state. To sum up, weight is the sum of reward obtained after the execution of this action multiplied discount and then reduced baseline. The gradient of $\bar{R}_\theta$ can be approximated as follows:

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \left(\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b \right) \nabla log \pi_\theta(a_t^n | s_t^n) \quad \gamma < 1 \tag{13}$$

This calculation of $\sum_t^{T_n} \left(\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b \right)$ is called advantage function $A^\theta(s_t, a_t)$ [Schulman, Moritz, Levine et al. (2015)], which means not that one action is absolutely good, but how good it is compared to others. The expectation of R_θ can be rewritten as the following:

$$\mathbb{E}_{(s_t, a_t) \sim \pi_\theta}[R_\theta] = \mathbb{E}_{(s_t, a_t) \sim \pi_\theta}\left[A^\theta(s_t, a_t) \nabla log \pi_\theta(a_t^n | s_t^n) \right] \tag{14}$$

2.4.3 Introduction to proximal policy optimization (PPO)

The traditional Policy Gradient method can only make an update with the samples obtained by sampling, then resampling and update the parameter. The method of proximal policy optimization proposed in 2017 [Schulman, Wolski, Dhariwal et al. (2017)] enables multiple epochs of minibatch updates, have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement. This method adopts the idea of importance sampling, assuming that it is impossible to select sample x from distribution π. One idea is to select alternative samples from another distribution q and multiply by a weight to correct the difference between the two distributions. The expected output $f(x)$ of x sampled from the distribution π approximate to the following formula:

$$\mathbb{E}_{x \sim p}[f(x)] = \mathbb{E}_{x \sim p}\left[f(x) \frac{p(x)}{q(x)} \right] \tag{15}$$

In general, we use π_θ to collect data. When θ is updated, we have to sample training data again. The goal of PPO is using the sample from $\pi_{\theta_{old}}$ to train θ. θ_{old} is fixed, so we can

re-use the sample data. Applying the idea of importance sampling to PPO, we can assume that there are two neural networks with parameters θ and θ_{old}. Then we sample the data to train θ many times from θ_{old}. The algorithm of gradient for update is shown as follows.

$$\mathbb{E}_{(s_t,a_t)\sim\pi_\theta}\left[A^\theta(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)\right]$$
$$= \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{old}}}\left[\frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_{old}}(s_t,a_t)}A^{\theta_{old}}(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)\right]$$
$$= \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a_t|\,s_t)}{\pi_{\theta_{old}}(a_t|\,s_t)}\frac{\pi_\theta(s_t)}{\pi_{\theta_{old}}(s_t)}A^{\theta_{old}}(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)\right] \quad (16)$$

Here, θ_{old} is the vector of policy parameters before the update. The new objective function $J^{\theta_{old}}(\theta)$ is as follows:

$$J^{\theta_{old}}(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a_t|\,s_t)}{\pi_{\theta_{old}}(a_t|\,s_t)}A^{\theta_{old}}(s_t,a_t)\right] \quad (17)$$

In TRPO [Schulman, Levine, Abbeel et al. (2015)], an objective function is maximized subject to a constraint on the size of the policy update. The new objective function of TRPO is as follows:

$$J^{\theta^k}_{\text{TRPO}}(\theta) = \max_\theta \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta^k}}\left[\frac{\pi_\theta(a_t|\,s_t)}{\pi_{\theta^k}(a_t|\,s_t)}A^{\theta^k}(s_t,a_t) - \beta KL\left[\pi_{\theta^k}(\cdot|s_t),\pi_\theta(\cdot\,|\,s_t)\right]\right] \quad (18)$$

TRPO adds a constraint that requires the KL divergence between the front and back policies to be less than a certain threshold. In the experiments of Schulman et al. [Schulman, Wolski, Dhariwal et al. (2017)] show that it is not sufficient to simply choose a fixed penalty coefficient β and optimize the penalized objective Eq. (18) with SGD; additional modifications are required. PPO uses the clip function to punish policy that keep $\frac{\pi_\theta(a_t|\,s_t)}{\pi_{\theta_{old}}(a_t|\,s_t)}$ away from 1, which indicates that the strategy $\pi_\theta(a_t|\,s_t)$ and $\pi_{\theta_{old}}(a_t|\,s_t)$ are quite different. The objective function of PPO is as follows:

$$J^{\theta^k}_{\text{PPO}}(\theta) = \sum_{(a_t,s_t)} min\frac{\pi_\theta(a_t|\,s_t)}{\pi_{\theta^k}(a_t|\,s_t)}A^{\theta^k}(s_t,a_t), clip\left(\frac{\pi_\theta(a_t|\,s_t)}{\pi_{\theta^k}(a_t|\,s_t)},1-\varepsilon,1+\varepsilon\right)A^{\theta^k}(s_t,a_t) \quad (19)$$

A proximal policy optimization (PPO) algorithm that uses fixed-length trajectory segments is shown in Algorithm 2. Each iteration, actor collect T steps of data in each of N iteration.

Algorithm 2 PPO, adopt from Schulman et al. [Schulman, Wolski, Dhariwal et al. (2017)]

for iteration = 1, 2, … **do**
  for actor = 1, 2, …, N **do**
    run policy $\boldsymbol{\pi_{\theta_{old}}}$ in environment for T steps
    compute advantage estimates $\boldsymbol{\hat{A}_1, \ldots, \hat{A}_T}$
  end for
  optimize surrogate $\boldsymbol{L}$ $\boldsymbol{wrt}$ $\boldsymbol{\theta}$, with $\boldsymbol{K}$ epochs and minibatch size $\boldsymbol{M \leq NT}$
  $\boldsymbol{\theta_{old} \leftarrow \theta}$
end for

2.4.4 Comparative introduction of various algorithms

Table 1: Comparative introduction of various algorithms

Value-based DRL	Deep Q-network	a) DQN uses an experience replay mechanism during the training process. Small batches are randomly selected from the memory unit, and the network parameters θ are updated using the SGD algorithm. b) Use a deep convolution network to approximate the current value function and another network to generate the target Q value.
	Deep double Q-network [Van Hasselt, Guez and Silver (2016); Hasselt (2010)]	a) There are two different sets of parameters in the double Q-network, separating the action selection from the strategy evaluation. b) Use the present value network with parameter θ to select the optimal action and use the target value network with parameter θ' to evaluate the optimal action. The objective Q-value as follows: $Y_i^{DDQN} = r + \gamma Q\left(s', \underset{a}{\operatorname{argmax}}\, Q(s,a|\theta_i)|\theta_i'\right)$
	Deep Q-network based on advanced learning	a) The sample error based on sampling is defined as: $\Delta\, Q(s,a) = r + \gamma V(s') - Q(s,a)$ b) Apply the two new operators defined by the advanced learning [Baird (1999)] to the AL error term and the PAL error term in Equation: $\Delta_{AL}\, Q(s,a) = \Delta\, Q(s,a) - \alpha[V(s) - Q(s,a)]$ $\Delta_{AL}\, Q'(s,a) = \Delta\, Q(s,a) - \alpha[V(s') - Q(s',a)]$ $\Delta_{PAL}\, Q(s,a) = \max\{\Delta_{AL}\, Q(s,a), \Delta_{AL}\, Q'(s,a)\}$ c) The difference of value functions between the optimal and suboptimal action is greater, and the Q value is more accurate.
	Prioritized Experience Replay [Schaul, Quan, Antonoglou et al. (2015)]	a) This method replaces uniform sampling with priority-based sampling to improve the sampling probability of some valuable samples. b) The temporal difference of each sample is used as a criterion for evaluating the priority. c) Two techniques for using stochastic prioritization and importance-sampling weights in the sampling process.
	Dynamic Frame Skip Deep Q-Network (DFDQN) [Lakshminarayanan, Sharma and Ravindran (2016)]	Use dynamic frame skipping to replace the action repeating k times at each moment in DQN, achieving better performance.
	Dueling DQN [Wang, Schaul, Hessel et al. (2015)]	a) The abstract features extracted by CNN are shunted into two branches, one representing the state value function and the other representing the state-based action advantage function. b) State value function is expressed as $\hat{V}(s\|\theta,\beta)$ and action advantage function is expressed as $\hat{A}(s,a\|\theta,\alpha)$. Combine state value flow with action advantage flow through an aggregation operation: $Q(s,a\|\theta,\alpha,\beta) = \hat{V}(s\|\theta,\beta) + \hat{A}(s,a\|\theta,\alpha)$
	Deep Recurrent Q-Network (DRQN) [Hausknecht and Stone (2015)]	Replace the first fully connected layer in the DQN with 256 LSTMs. At this time, the input of the model is only one image at the current time, instead of the four images in DQN, which reduces the computational resources consumed by the deep network perception image features.

Policy-based DRL	Actor-Critic (AC) [Bahdanau, Brakel, Xu et al. (2016)]	Stochastic Actor-Critic [Bhatnagar and Kumar (2004)]	a) Stochastic Policy Gradient Theorem $\nabla_\theta J(\pi_\theta) = \int_S \rho^\pi(s) \int_A \nabla_\theta \pi_\theta(a\|s) Q^\pi(s,a) da ds = \mathbb{E}_{S\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta log\pi_\theta(a\|s) Q^\pi(s,a)]$ b) The actor uses stochastic gradient ascent to update parameter θ of the stochastic strategy $\pi_\theta(s)$. Although the real value function $Q^\pi(s,a)$ is not known, the approximate value function $Q^w(s,a)$ is created using the parameter w.Try to make $Q^\pi(s,a) \approx Q^w(s,a)$ with the appropriate strategy gradient algorithm: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{S\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta log\pi_\theta(a\|s) Q^w(s,a)]$
		Off-policy Actor-Critic	$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{S\sim\rho^\beta, a\sim\beta}\left[\frac{\pi_\theta(a\|s)}{\beta_\theta(a\|s)} \nabla_\theta log\pi_\theta(a\|s) Q^\pi(s,a)\right]$
		Deterministic Policy Gradient (DPG) [Silver, Lever, Heess et al. (2014)]	a) If the MDP satisfies $p(s'\|s,a)$, $\nabla_a p(s'\|s,a)$, $\mu_\theta(s)$, $\nabla_\theta \mu_\theta(s), r(s,a)$, $\nabla_a r(s,a), p_1(s)$ being continuous under parameters s, a, s', x, then it means that $\nabla_\theta \mu_\theta(s)$ and $\nabla_a Q^\mu(s,a)$ exist and the deterministic strategy gradient exists: $\nabla_\theta J(\mu_\theta) = \int_S \rho^\pi(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a) \|a = \mu_\theta(s) ds = \mathbb{E}_{S\sim\rho^\pi}[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)]\|a = \mu_\theta(s)]$ b) Deterministic strategy is a special case of random strategy gradient.
		Deep Deterministic Policy Gradient (DDPG) [Usunier, Synnaeve, Lin et al. (2016)]	a) This algorithm is used to solve the DRL problem in continuous action space. b) Deterministic strategy $a = \pi(s\|\theta^\mu)$ and value function $Q(s,a\|\theta^Q)$ are represented using depth neural networks with parameters θ^μ and θ^Q. c) $\nabla_{\theta^\mu} J(\theta^\mu) = \mathbb{E}_S[\nabla_a Q(s,a\|\theta^Q) \nabla_{\theta^\mu} \pi(s\|\theta^\mu)]$ $\nabla_{\theta^Q} J(\theta^Q) = \mathbb{E}_{s,a,s'\sim D}\left[\left(y - Q(s,a\|\theta^Q)\right) \nabla_{\theta^Q} Q(s,a\|\theta^Q)\right]$ $y = r + \gamma Q(s', \pi(s'\|\theta^\mu)\theta^Q)$
		Stochastic Value Gradient (SVG) [Heess, Wayne, Silver et al. (2015)]	Using parametric mathematical techniques to learn the generation model of environmental dynamics, the deterministic policy gradient method is extended to a strategy optimization process in a random environment.
	Asynchronous Advantage Actor-Critic (A3C) [Mnih, Badia, Mirza et al. (2016)]	a) Execute multiple agents in parallel and asynchronously using CPU multithreading. b) Parallel agents go through many different states at any time, removing the correlation between state transition samples generated during training.	
	Advantage Actor-Critic (A2C) [Kuutti, Bowden, Joshi et al. (2019)]	Change asynchronous to synchronous, better use of GPU, good effect when batch-size is large	

3 Image processing using deep reinforcement learning

Many methods of image processing in deep learning have been introduced. Convolutional neural network uses a special structure for image recognition, which can be trained quickly and accuracy. Because of its fast speed, it is easy to adopt multi-layer neural network, and the multi-layer structure has a great advantage in the accuracy of recognition, so the convolution neural network has made great achievements in image processing. However, it is very computationally expensive to apply convolutional neural network to the calculation of large images, because the calculation amount is linearly with the number of image pixels. In this section, we are going to introduce some of the applications of deep reinforcement learning in image processing mentioned in some papers.

3.1 Image classification

Mnih et al. [Mnih, Heess and Graves (2014)] proposed the recurrent attention model (RAM), which is a novel recurrent neural network model and can be trained using reinforcement learning methods. This model firstly selects a group of areas or position sequences through a random distribution, and only process the selected areas using glimpse sensor with high resolution, so as to achieve feature extraction of images or video information. Then, feature vectors and pictures are taken as the input of glimpse network which produce glimpse representation. Finally, the RNN model takes the glimpse representation and the internal representation at previous time step as input, produces the new internal state which produces the next location and action. The above iteration is repeated in the RNN model. At each glimpse the agent takes, the environment will give it a negative reward to force it to trade off making correct classifications with the cost of taking more glimpses.

Ba et al. [Ba, Mnih and Kavukcuoglu (2014)] proposed the deep recurrent attention model (DRAM) which is used to recognize multiple objects in images and an improvement of the model proposed by Mnih's paper [Mnih, Heess and Graves (2014)]. This model is closer to the way humans process visual sequence tasks. It moves the fovea to the next relevant object or character and adds the identified target to our internal representation sequence. A multi-resolution image is cropped into a glimpse and input into the deep recursive neural network to update the internal representation and output the next glimpse location and the next object in the sequence. This process is iterated until the model does not need to process any objects. The training method in this paper can be used to identify multiple targets in a picture. The author has several experiments. First, the model learned to find the Numbers in the picture, and then two more challenging tasks were do addition and read house Numbers.

3.2 Face hallucination and face recognition

Cao et al. [Cao, Lin, Shi et al. (2017)] presented Attention-aware Face Hallucination (Attention-FH) based on deep reinforcement learning. Different from the traditional method, Attention-aware Face Hallucination combines the facial hallucination problem with the Markov decision-making problem. At each time step, the previous historical information is entered into the recurrent neural network to update the next input region. State can be exploited and explored by means of local enhancement network. Attention-FH trains recurrent policy network and local enhancement network by maximizing long-

term rewards, so that the local enhancement of the human face can be realized due to the local correlation of the images.

Rao et al. [Rao, Lu and Zhou (2017)] proposed an attention-aware deep reinforcement learning (ADRL) method for video face recognition, which is based on Markov decision process to eliminate misleading and confusing frames in video. The model input image space and feature space into the convolution neural network for spatial representation learning, which can make better use of the discarded face information in the process of feature learning. Next the output after learning is input into the local recurrent network and local temporal pooling for temporal representation learning. Then the output is input into the frame evaluation network for attention-aware reinforcement learning, which finds the attentions of the video pair for face verification.

3.3 Active object localization and visual object tracking

Caicedo et al. [Caicedo and Lazebnik (2015)] proposed an active detection model for locating object in scene, which learns to determine the most specific location of the object through a top-down search strategy. Firstly, the model uses a pre-trained CNN as a feed-forward feature extractor so as learning the Q function is faster. Then the model uses deep reinforcement learning to train the location agent, which uses simple transformation operations to refine the size and geometry of a bounding box. The policy followed during training is $\varepsilon - greedy$. First analyze the entire scene and then narrow down the location. Each transformation should keep the object in the visible area while minimizing the background. State is the picture after each localization. The agent chooses action to further locate the object based on the observation of state, so as to achieve localization of the target. Basing on the deep q-network algorithm, a reward function is set according to the degree of the current box covering the object to learn a positioning strategy. This method has a good result under the data set Pascal Voc.

Zhang et al. [Zhang, Maei, Wang et al. (2017)] proposes a completely end-to-end and fully off-line method for visual tracking in video based on their findings that tracking problems can be thought of as a continuous decision-making process. The model learns to predict the position of the target object in the bounding box of each frame and to learn good tracking policies that pay attention to continuous, inter-frame correlation. The model is an agent integrates convolutional neural network with recurrent convolutional neural network that interacts with a video overtime, and it can be trained with reinforcement learning (RL) algorithms to maximize tracking performance in the long run.

Guo et al. [Guo, Lu and Zhou (2018)] proposed a dual-agent deep reinforcement learning (DADRL) method for deformable face tracking. Since the performance of facial landmarks largely depends on the accuracy of the generated bounding box, the author proposed the unified framework for simultaneous bounding box tracking and landmark detection based on the interaction between the two tasks of generating bounding box and detecting face landmarks, and learned the two conditional distributions at the same time. Two agents following Markov decision process are adopted to complete these two tasks, and messages are transmitted through adaptive action sequence under the framework of deep reinforcement learning, so as to update the location of boundary box and facial landmark iteratively.

Ren et al. [Ren, Yuan, Lu et al. (2018)] proposed a deep reinforcement learning method based on iterative shift (DRL-IS) method for motion estimation and tracking state change. This method predicts the iterative shift of the target boundary box in an end-to-end manner by introducing an actor-critic network, and evaluates the shift to update the target model.

Merkos [Merkos (2019)] proposed an end-to-end deep reinforcement learning (DRL) method PPO algorithm for video tracking to predict the position of the target object in the bounding box of each frame. The model adopts actor-critic structure and is composed of two neural network structures: one is the action decision (policy) network used to generate actions; Another is a network of critics that evaluates value functions and explores the state space. The two neural networks jointly train and use a Perceptual hashing algorithm dhash to allocate rewards for agents, and to achieve better tracking performance by maximizing rewards.

3.4 Image segmentation

Abtahi et al. [Abtahi, Zhu and Burry (2015)] presented a method of license plate character segmentation based on deep reinforcement learning, and applied Deep RL to improve the character segmentation unit of the ALPR system. A segmentation agent is established to find the segmentation path which are valid and which are not valid. The segmentation path result is come from an aggressive projection segmentation step. The paper proposed a hybrid approach through combining the speed and simplicity of project-based segmentation and the power of RL approach, thus improving the correct rate of segmentation. The experimental results show that this method has obvious advantages over the histogram projection method.

Ghajari et al. [Ghajari, Bagher and Sistani (2017)] applying reinforcement learning to the segmentation of ultrasonic images and the improvement of segmentation quality. The method can be divided into three stages: pre-processing, processing and post-processing. Pre-processing uses multi-agent dimensional structure to select the appropriate sub-image size and divide the image into a group of sub-images. In the processing stage, state, action and reward are introduced to further segment the segmented image as the input. In the post-processing stage, trial-and-error method is adopted to improve the image segmentation quality.

Song et al. [Song, Myeong and Lee (2018)] proposes an automatic seed generation technology based on deep reinforcement learning to solve the problem of interactive segmentation. The author transformed the automatic seed generation problem into Markov decision process (MDP) and trained the seed generation agent with deep reinforcement learning. Then it is optimized by deep q-network (DQN). Since the image segmentation technology needs to be carried out in the context of understanding the user's intention, the technology first simulates human to describe image and uses it as input information, which can be a graffiti or boundary box. Then it interacts with the segmentation system to obtain the desired object. The artificial user enters a point on the desired object and a point on the background, and the system will automatically locate the sequence that the user is interested in. Finally, a new foreground or background seed is determined as the new input according to the output of the system.

3.5 Image enhancement

Park et al. [Park, Lee, Yoo et al. (2018)] presented a deep reinforcement learning to enhance color without the intermediate process supervision. Inspired by the modification process of human beings, the task of enhancing image color is completed by learning the modification process step by step. This method adopts the training scheme of "distortion-recovery", which does not need to input and modify images pairs to train only high-quality reference images. The problem of insufficient training samples was solved by distorting the training pictures.

Yu et al. [Yu, Liu, Zhang et al. (2018)] proposes a new algorithm for learning local exposure by deep reinforcement adversarial learning. In reinforcement learning, we divide the image into sub-images and reflect the dynamic exposure changes of these sub-images according to the original low-level characteristics. Use the policy gradient to learn sequentially each sub-image to maximize the global reward function and learn multiple local exposure operations. The original input image is retouched with each training result to obtain multiple retouched images, which are finally mixed with the final trained exposure. In generative antagonistic network, discriminator is used as the value function of reinforcement learning.

3.6 Image captioning

Ren et al. [Ren, Wang, Zhang et al. (2017)] proposes a decision-based framework to complete the image captioning task, which is different from the previous encoder-decoder framework. This is an image captioning method based on deep reinforcement learning, and it is also the first attempt to apply the decision-based framework to the image captioning. The reward function based on visual semantic embedding is also introduced in this method. This method has achieved its best performance by the existing standard benchmark.

3.7 Image restoration

Yu et al. [Yu, Dong, Lin et al. (2018)] proposes the toolchain concept in an innovative way. For each picture to be repaired, various tools can be selected from the tool box to deal with specific problems. These tools process pictures in a certain order, which forms a tool chain. Each tool in the toolbox is a lightweight convolutional neural network that is used to solve a specific problem and implement a single function such as denoising or deblurring. The author regards the selection of the tool sequence as a Markov decision process and solves it with deep reinforcement learning. In addition, in order to solve the new distorted results that may be produced by the previous recovery operation, the authors adopted a method of joint training tools and reinforcement learning agents.

3.8 Summary

Table 2: The summary of the literatures of image processing using deep reinforcement learning

	Literatures	Brief summary of model	Advantage and disadvantage
Image classif ication	Mnih et al. [Mnih, Heess and Graves (2014)]	the agent of this deep reinforcement learning network is constructed by a recursive neural network. At	Advantage: Dealing with interactive problems with visual input is as easy as dealing with images.

		each time step, the next action and new position are selected through the input features and positions and get reward from environment.	Disadvantage: Difficult to deal with multiple objects in one image.
	Ba et al. [Ba, Mnih and Kavukcuoglu (2014)]	The proposed model is a deep recurrent neural network trained with reinforcement learning to process only a glimpse of an image which is the most relevant region.	Advantage: a) recognize multiple objects purely from label sequences. b) using both fewer parameters and much less computation and more accurate compared to CNN. c) Different sizes of images are processed at the same computational cost. d) DRAM is less likely to overfit than CNN. e) can easily deal with variable length label sequences
Face hallucination and face recognition	Cao et al. [Cao, Lin, Shi et al. (2017)]	The author proposed a novel attention-aware Face Hallucination (attention-FH) framework, which uses deep reinforcement learning to achieve facial enhancement by making fully use of the interdependency between the sequential patches of the image.	Advantage: a) This method has the most advanced performance in mainstream data sets such as BioID and LFW. b) FH can determine the optimal search path based on the features of each face image
	Rao et al. [Rao, Lu and Zhou (2017)]	The model trains the attention model through the deep reinforcement learning framework without additional labels.	Advantage: a) recognize different pairs of videos through different attentions on video. b) The model can also be applied in other computer vision tasks.
	Wang et al [Wang, Lin, Chao et al. (2017)]	Face feature extraction, transformation and comparison based on deep reinforcement learning and convolution neural network are studied to realize face recognition.	Advantage: The method can detect dynamic face images, which are often blurred and affected by the details of different facial expressions and changes in lighting.
Active object localization and visual object tracking	Caicedo et al. [Caicedo and Lazebnik (2015)]	The model uses an active detection model based on Deep Q Network algorithm for localizing objects in scenes. And the author proposed a dynamic attention action strategy to focus on objects while transform box.	Advantage: a) Use different search paths for different objects in different scenarios. b) It is a very efficient strategy for applications where a few numbers of categories are required.

	Zhang et al. [Zhang, Maei, Wang et al. (2017)]	The author proposes a novel framework, referred to as Deep RL Tracker (DRLT), which integrates convolutional network with recurrent network and processes video frames as a whole. The purpose of the model is directly outputs location predictions of the target in each frame.	Advantage: a) Time correlation in video is explicitly exploited by introducing RNN and RL. b) The most advanced performance is achieved on the OTB public tracking benchmark.
	Guo et al. [Guo, Lu , Zhou et al. (2018)]	The model uses the bayesian model of Dual-agent deep reinforcement learning (DADRL) to realize the probabilistic interaction between these two processes, simultaneous bounding box tracking and landmark detection	Advantage: a) learned the two conditional distributions at the same time. b) DADRL is improved greatly compared with the most advanced shape-able face tracking method on 300-vw data set.
	Ren et al. [Ren, Yuan, Lu et al. (2018)]	The author proposed a DRL-IS method for visual tracking.	Advantage: a) This method has strong robustness and can deal with large deformation and sudden movement b) Very computationally efficient
	Merkos et al. [Merkos (2019)]	The model is a neural network tracking model based on Convolutional network trained with deep reinforcement learning PPO algorithm using Actor-Critic model.	Disadvantage: a) The algorithm performs poorly in low illumination video b) The border box cannot change the shadow on the width and height.
Image segmentation	Abtahi et al. [Abtahi, Zhu and Burry (2015)]	Adopting RL to establish a segmentation agent, which can find a suitable segmentation path from top to bottom in the cropped license plate image to avoid cutting characters.	Advantage: Compared with the existing histogram projection method, this method is obviously improved
	Song et al. [Song, Myeong and Mu Lee (2018)]	An interactive segmentation technology using the Deep Q-Network optimization system. When the user specifies separately a point on the desired object and background, a human user input sequence is automatically generated to accurately segment the target object.	Advantage: a) Agents can predict users' intentions so significantly reduce user input can still achieve good results. b) The agent can help to reduce the cost of pixelwise labeling task.

Image enhancement	Park et al. [Park, Lee, Yoo et al. (2018)]	A deep reinforcement learning method is proposed to simulate the process of human image editing. Adopt "distortion – recovery" training program.	Advantage: It has been verified that this method performs better on MIT-Adobe FiveK data set than supervised learning methods such as Pix2Pix.
	Yu et al. [Yu, Liu, Zhang et al. (2018)]	The author proposes a novel deep reinforcement adversarial learning algorithm, which learns the optimal exposure operations of retouching low-quality images.	Advantage: a) The framework can enhance local areas of images flexibly with only exposure operation. b) Learning process is stable. c) Can well preserve the details of the raw image. d) Using discriminator as value function can reduce memory consumption and improve training speed

4 Conclusion

In this paper, we first introduce the background of deep reinforcement learning. Based on the implementation process of reinforcement learning algorithm, the methods and two algorithms of reinforcement learning are discussed in detail. On this basis, the deficiencies of reinforcement learning are proposed, and deep reinforcement learning is introduced to make up for the deficiencies. Then, we briefly discuss the development of deep reinforcement learning in recent years and summarizes and compares different deep reinforcement learning algorithms, focusing on introducing three basic algorithms: Deep Q-Network, policy gradient and PPO. Finally, according to the application of deep reinforcement learning in different aspects of image processing, such as image classification, face hallucination and face recognition, active object localization and visual object tracking, image segmentation, image enhancement, image recovery. We analyze the deep reinforcement learning methods used in different papers and compare their advantages and disadvantages.

Funding Statement: This work was supported in part by the Open Research Project of State Key Laboratory of Novel Software Technology under Grant KFKT2018B23, the Priority Academic Program Development of Jiangsu Higher Education Institutions, the 2018 Tiancheng Huizhi Innovation Promotion Education and Scientific Research Innovation Fund of the Ministry of Education under Grant 2018A03038 and the Open Project Program of the State Key Lab of CAD&CG (Grant No. A1916), Zhejiang University.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Abtahi, F.; Zhu, Z.; Burry, A. M. (2015): A deep reinforcement learning approach to character segmentation of license plate images. *14th IAPR International Conference on Machine Vision Applications*, Tokyo, pp. 539-542.

Baird, L. C. (1999): Reinforcement Learning Through Gradient Descent (Ph.D. Thesis). *Carnegie Mellon University*, Pittsburgh.

Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R. et al. (2016): An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Bhatnagar, S.; Kumar, S. (2004): A simultaneous perturbation stochastic approximation-based actor-critic algorithm for Markov decision processes. *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 592-598.

Ba, J.; Mnih, V.; Kavukcuoglu, K. (2014): Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*.

Cao, Q.; Lin, L.; Shi, Y.; Liang, X.; Li, G. (2017): Attention-aware face hallucination via deep reinforcement learning. *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, pp. 690-698.

Caicedo, J. C.; Lazebnik, S. (2015): Active object localization with deep reinforcement learning. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2488-2496.

Fang, W.; Zhang, F.; Sheng, V. S.; Ding, Y. (2018): A method for improving CNN-based image recognition using DCGAN. *Computers, Materials & Continua*, vol. 57, no. 1, pp. 167-178.

Fang, W.; Zhang, F.; Ding, Y.; Sheng, V. S. (2020): A new sequential image prediction method based on LSTM and DCGAN. *Computers, Materials & Continua*, vol. 64, no. 1, pp. 217-231.

Goodfellow, I.; Bengio, Y.; Courville, A. (2016): *Deep Learning*. MIT Press.

Guo, M.; Lu, J.; Zhou, J. (2018): Dual-agent deep reinforcement learning for deformable face tracking. *European Conference on Computer Vision*, pp. 783-799.

Ghajari, S.; Bagher, M.; Sistani, N. (2017): Improving the quality of image segmentation in Ultrasound images using Reinforcement Learning. *Communications on Advanced Computational Science with Applications*, pp. 33-40.

Hinton, G. E.; Osindero, S.; Teh, Y. W. (2006): A fast learning algorithm for deep belief nets. *Neural computation*, vol. 18, no. 7, pp. 1527-1554.

Hasselt, H. V. (2010): Double Q-learning. *Advances in Neural Information Processing Systems*, pp. 2613-2621.

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

Heess, N.; Wayne, G.; Silver, D.; Lillicrap, T.; Erez, T. et al. (2015): Learning continuous control policies by stochastic value gradients. *Neural Information Processing Systems*, pp. 2944-2952.

Kaelbling, L. P.; Littman, M. L.; Moore, A. W. (1996): Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285.

Kuutti, S.; Bowden, R.; Joshi, H.; de Temple, R.; Fallah, S. (2019): End-to-end reinforcement learning for autonomous longitudinal control using advantage actor critic with temporal context. *IEEE Intelligent Transportation Systems Conference*, pp. 2456-2462.

LeCun, Y.; Bengio, Y.; Hinton, G. (2015): Deep learning. *Nature*, vol. 521, no. 7553, pp. 436-444.

Lakshminarayanan, A. S.; Sharma, S.; Ravindran, B. (2016): Dynamic frame skip deep q network. *arXiv preprint arXiv:1605.05365*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J. et al. (2015): Human-level control through deep reinforcement learning. *Nature*, vol. 518, no. 7540, pp. 529.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I. et al. (2013): Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Heess, N.; Graves, A. (2014): Recurrent models of visual attention. *Neural Information Processing Systems*, pp. 2204-2212.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. et al. (2016): Asynchronous methods for deep reinforcement learning. *International Conference on Machine Learning,* pp. 1928-1937.

Merkos, A. (2019). *An Actor-Critic Deep Reinforcement Learning agent for Visual Object Tracking (Ph.D. Thesis)*. *University of Ioannina*.

Park, J.; Lee, J. Y.; Yoo, D.; So Kweon, I. (2018): Distort-and-recover: Color enhancement using deep reinforcement learning. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5928-5936.

Rao, Y.; Lu, J.; Zhou, J. (2017): Attention-aware deep reinforcement learning for video face recognition. *IEEE International Conference on Computer Vision*, pp. 3931-3940.

Ren, L.; Yuan, X.; Lu, J.; Yang, M.; Zhou, J. (2018): Deep reinforcement learning with iterative shift for visual tracking. *European Conference on Computer Vision*, pp. 684-700.

Ren, Z.; Wang, X.; Zhang, N.; Lv, X.; Li, L. J. (2017): Deep reinforcement learning-based image captioning with embedding reward. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 290-298.

Sutton, R. S.; Barto, A. G. (2018): *Reinforcement Learning: An Introduction.* MIT Press.

Sutton, R. S. (1988): Learning to predict by the methods of temporal differences. *Machine learning*, vol. 3, no. 1, pp. 9-44.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. (2015): High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438.*

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. (2015): Trust region policy optimization. *International Conference on Machine Learning*, pp. 1889-1897.

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

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

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D. et al. (2014): Deterministic policy gradient algorithms. *International Conference on Machine Learning.*

Song, G.; Myeong, H.; Mu Lee, K. (2018): Seednet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1760-1768.

Usunier, N.; Synnaeve, G.; Lin, Z.; Chintala, S. (2016): Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993*.

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

Watkins, C. J. C. H. (1989): *Learning from Delayed Rewards (Ph.D. Thesis)*. King's College, Oxford.

Watkins, C. J.; Dayan, P. (1992): Q-learning. *Machine Learning*, vol. 8, no. 3-4, pp. 279-292.

Williams, R. J. (1992): Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, vol. 8, no. 3-4, pp. 229-256.

Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M. et al. (2015): Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

Wang, P.; Lin, W. H.; Chao, K. M.; Lo, C. C. (2017): A face-recognition approach using deep reinforcement learning approach for user authentication. *IEEE 14th International Conference on e-Business Engineering*, pp. 183-188.

Yu, R.; Liu, W.; Zhang, Y.; Qu, Z.; Zhao, D. et al. (2018): Deepexposure: Learning to expose photos with asynchronously reinforced adversarial learning. *Neural Information Processing Systems*, pp. 2149-2159.

Yu, K.; Dong, C.; Lin, L.; Change Loy, C. (2018): Crafting a toolchain for image restoration by deep reinforcement learning. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2443-2452.

Zhang, D.; Maei, H.; Wang, X.; Wang, Y. F. (2017): Deep reinforcement learning for visual object tracking in videos. *arXiv preprint arXiv:1701.08936*.