Expert Systems With Applications 231 (2023) 120495

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Review

Reinforcement learning algorithms: A brief survey

Ashish Kumar Shakya [1], Gopinatha Pillai [2], Sohom Chakrabarty [3]

Department of Electrical Engineering, Indian Institute of Technology, Roorkee, Uttarakhand 247667, India

ARTICLE INFO

Keywords:
Reinforcement learning
Stochastic optimal control
Function approximation
Deep Reinforcement Learning (DRL)

ABSTRACT

Reinforcement Learning (RL) is a machine learning (ML) technique to learn sequential decision-making in complex problems. RL is inspired by trial-and-error based human/animal learning. It can learn an optimal policy autonomously with knowledge obtained by continuous interaction with a stochastic dynamical environment. Problems considered virtually impossible to solve, such as learning to play video games just from pixel information, are now successfully solved using deep reinforcement learning. Without human intervention, RL agents can surpass human performance in challenging tasks. This review gives a broad overview of RL, covering its fundamental principles, essential methods, and illustrative applications. The authors aim to develop an initial reference point for researchers commencing their research work in RL. In this review, the authors cover some fundamental model-free RL algorithms and pathbreaking function approximation-based deep RL (DRL) algorithms for complex uncertain tasks with continuous action and state spaces, making RL useful in various interdisciplinary fields. This article also provides a brief review of model-based and multi-agent RL approaches. Finally, some promising research directions for RL are briefly presented.

1. Introduction

Reinforcement Learning (RL) is a learning approach in which an artificial intelligence (AI) agent interacts with its surrounding environment by trial-and-error method and learns an optimal behavioral strategy based on the reward signals received from previous interactions. This learning process of the RL agent mimics the human/animal learning approach. One of the most fruitful aspects of RL is its adaptability to other scientific and engineering fields. RL has emerged as an efficient technique for solving complicated sequential decision-making tasks in recent years. It offers a great opportunity to open new frontiers in technology where system models are unavailable or too difficult and/or expensive to build. The modern RL is the combination of three research fields, the psychology of human learning, the solution of optimal control using Dynamic Programming (DP), and temporal difference (TD) learning.

In 1911, Edward Thorndike gave the idea of the "Law of Effect," which was inspired by the trial-and-error learning principle (Thorndike, 1911). The basic idea of this law was that responses generating a pleasing impact in a situation are more likely to occur again in the same situation, while responses that cause discomfort are less likely to occur again in the same situation. Implementation of trial-and-error learning in a computer was a fascinating idea in artificial intelligence. In 1948, Alan Turing suggested a "pleasure-pain system" (Turing, 1948) which was based on the Law of Effect. In 1952, Claude Shannon's work on maze running mechanical mouse was one of the first successful demonstrations of trial-and-error learning (Shannon, 1952). Based on trial-and-error learning, Farley and Clark suggested a artificial neural network architecture for learning. The term "reinforcement" was first introduced in Pavlov's work on conditioned reflexes (Pavlov & Anrep, 1927) and was followed in the 1960s (Minsky, 1961; Waltz & Fu, 1965; Mendel, 1966; Fu, 1970). In 1968, Michie and Chambers introduced a reinforcement learning agent, GLEE, and a controller called BOXES for the game tic-tac-toe (Michie & Chambers, 1968). The use of controller BOXES for a pole balancing task is one of the fundamental reference works in RL (Michie & Chambers, 1968). Harry Klopf's work in the 1970s in trial-and-error learning revived the field of RL (Klopf, 1972, 1975, 1982) and cleared most of the confusions about reinforcement and supervised learning. Sutton enhanced Klopf's animal learning theories in his initial work on brain theory (Sutton, 1978). Using Klopf's work and animal psychology theories, Sutton and Barto developed classical conditioning models based on TD learning (Sutton & Barto, 1981a, 1981b).

E-mail addresses: akumarshakya@ee.iitr.ac.in (A.K. Shakya), gn.pillai@ee.iitr.ac.in (G. Pillai), sohom.chakrabarty@ee.iitr.ac.in (S. Chakrabarty).
[1] 0000-0002-2541-7860
[2] 0000-0002-7386-5270
[3] 0000-0001-7213-6693

https://doi.org/10.1016/j.eswa.2023.120495
Received 25 January 2022; Received in revised form 22 April 2023; Accepted 12 May 2023
Available online 23 May 2023

Nomenclature

List of important symbols and notation

x, x', X	current state, next state, and state space
u, U	action, and action space
$U(x)$	set of possible actions in state x
r, R, G	step reward, reward space, and return
$\pi, \pi^*, \hat{\pi}$	policy, optimal policy, and approximate policy
k, K	time steps
γ, α, δ	discount factor, learning rate, and temporal difference
V, V_π	state value function and state value function for a policy
$V^*, \hat{V}$	optimal and approximate state value function
Q, Q_π	action value function and action value function for a policy
$Q^*, \hat{Q}$	optimal and approximate action value function
e, λ	eligibility trace, and eligibility parameter
Θ, ω	policy and value function approximation parameters
Θ^-, ω^-	the same, but for the target networks in DRL
$\mathscr{D}$	Experience or replay memory
$\nabla_\bullet$, $\sim$	gradient of a function w.r.t. some parameter, and sampling from a distribution
$\mathbb{E}[\bullet], p\{\bullet\}$	expectation, and probability

The term optimal control was first introduced in 1950. The aim of the optimal control theory was to find an optimal control law for a dynamical system such that it can minimize a cost function. Optimal control theory is mainly established on the concepts of calculus of variations and mathematical optimization. In the mid-1950, Richard Bellman gave one of the solutions to this problem with the Bellman equation (Bellman, 1956), which is based on states, value functions, and returns. Dynamic Programming (DP) method uses the Bellman equation to solve optimal control problems (Bellman, 1958, 1972). The discrete version of stochastic optimal control problems is known as the Markov decision process (MDP) (Puterman, 1994). Sequential decision-making is required for MDPs, which can suggest the control action for each visited state (Bellman, 1957). In 1960, Ronald Howard developed the policy iteration method for MDPs (Howard, 1960). All these methods are the roots of the modern RL theory and algorithms.

DP is an optimization technique that breaks down the optimization task into subtasks utilizing the idea of recursion to obtain the solution. Stochastic optimal control problems can be solved using the DP method, but the computation power required grows at an exponential rate as the number of states in the system increases. So the solution of optimal control problems by DP was not an efficient learning method. The offline computation of DP requires accurate system models, which are not available for most systems. In 1977, Paul Werbos introduced an approximate approach to DP, known as "heuristic dynamic programming" (Werbos, 1977). Then in 1989, Chris Watkins integrated the DP with online learning, which is known as Q learning (Watkins, 1989). Since then, these approaches have been extensively established by many researchers. In 1996, Dimitri Bertsekas and John Tsitsiklis introduced a concept based on DP and artificial neural networks, which is known as "neurodynamic programming" (Bertsekas & Tsitsiklis, 1996).

The other relevant research field associated with the history of RL is TD learning. Instead of calculating the total future return of an entire episode, TD learning predicts the values of the states with the help of visited states and immediate rewards. TD algorithm updates its prediction using the difference between new information and old prediction, which is known as "temporal difference". In 1959, Arthur Samuel used a TD-based learning method for his computer checkers program (Samuel, 1959), that was inspired by Claude Shannon's Chess-playing computer program (Shannon, 1950). In 1977, Ian Witten introduced the idea of TD (0), which significantly contributed to the field of TD learning (Witten, 1977). TD(0) is the simplest form of TD learning, where prediction values are updated after each state transition.

Generally, MDPs and their solutions are limited to systems with discrete action and state spaces. Function approximation techniques overcome these limitations. In recent years for various applications, deep learning (DL) successfully outperformed traditional ML approaches in terms of efficiency and accuracy. Deep neural networks (DNN) have been quite effective in various fields, including computer vision, speech recognition, video games, board games programs, natural language processing, and robotics, producing excellent results (Simonyan & Zisserman, 2015; Karpathy et al., 2014). Modern RL research is mainly based on the function approximation by DNN, referred to as Deep Reinforcement Learning (DRL). Google DeepMind's success with problems like Atari Games (Mnih et al., 2013; Mnih et al., 2015), and AlphaGo (Silver et al., 2016; Silver et al., 2017a; Silver et al., 2018; Schrittwieser et al., 2020) has revolutionized the field of DRL. After the successes of the IBM chess-playing program Deep Blue in 1997 (Campbell et al., 2002) against the chess world champion Garry Kasparov and TD learning-based backgammon computer program TD Gammon in 1992 by Gerald Tesauro (Tesauro, 1995), DeepMind's success in game playing was a remarkable achievement in the field of artificial intelligence (AI). In ML, generalization refers to the ability of an algorithm to be effective across a range of new inputs and applications. DRL is a powerful technique for model-free learning, which also satisfies the property of generality. Thereby, without the knowledge of complete system dynamics, DRL can learn directly from experience samples in offline or online modes. Thus DRL is an efficient technique to find a near-optimal policy for stochastic nonlinear systems having continuous state and action spaces. DRL can lead us to the construction of an autonomous AI agent with superhuman performance (Li et al., 2019) and can solve some of the most challenging optimization problems in various fields using policy gradient (Sutton et al., 2000), actor-critic architectures (Crites & Barto, 1994; Konda & Tsitsiklis, 2000; Konda & Tsitsiklis, 2003; Sutton et al., 2000) and policy optimization methods (Schulman et al., 2015, 2017).

Currently, DRL algorithms are being applied in the field of robotics to learn optimal control policies directly from visual inputs for different real-world problems (Kober et al., 2013; Polydoros & Nalpantidis, 2017). Researchers from various domains, including AI, control, robotics (García & Shafie, 2020; Pane et al., 2019; Kobayashi & Sugino, 2020; Duguleana & Mogan, 2016; Miljković et al., 2013), intelligent vehicles (De Morais et al., 2020), economics (Wu et al., 2020b; Soleymani & Paquet, 2021), medicine, and others (Yin et al., 2021; Yu et al., 2021; Zhang et al., 2022), have contributed to RL. As a result, several survey papers about the important development and application of RL in different fields have been published (Li et al., 2019; Nguyen et al., 2017; Buşoniu et al., 2018; Liu et al., 1907; Arulkumaran et al., 2017; Luong et al., 2019; Xu et al., 2014). Based on the RL literature that is currently available, the major RL milestones are depicted in Fig. 1.

Table 1 lists some of the influential works that have shaped contemporary RL research.

1.1. What distinguishes this survey from the ones that came before it?

In the past few years, a large number of survey papers on reinforcement learning have been published. However, the majority of these studies discuss the applications of RL in particular domains, such as industrial process control (Nian et al., 2020), autonomous vehicles (Kiran et al., 2022; Aradi, 2022), robotics (Singh et al., 2022; Zhu & Zhang, 2021; Khan et al., 2020), traffic signal control (Noaeen et al., 2022; Rasheed et al., 2020), energy management (Chen et al., 2022; Fu et al., 2022; Arwa & Folly 2020), recommender systems (Afsar et al., 2022), computer vision (Le et al., 2022), medical (Zhou et al., 2021; Subramanian et al., 2022), etc. In Arulkumaran et al. (2017), the fundamental concepts of key deep reinforcement learning approaches,

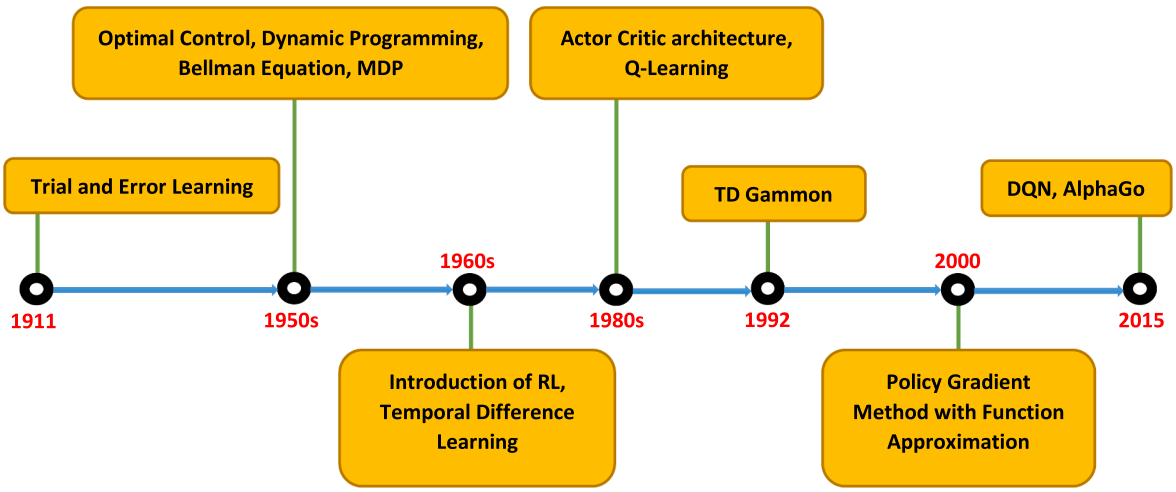

Fig. 1. Milestones of reinforcement learning.

such as the deep Q-network, policy gradient, and actor-critic, are briefly explained. The survey (Arulkumaran et al., 2017) also discusses a number of recent research areas in the field of RL. The survey paper Li (2018) provides a fantastic overview of RL methods up to 2018. This article goes into considerable detail about a number of key components and significant mechanisms of RL, along with applications and ongoing work in DRL. The recent survey papers by Moerland et al. (2022) and Luo et al. (2022a) give a detailed review of model-based RL approaches.

There are additional noteworthy survey studies that provide an excellent summary of some of the research directions in RL. The Multi-agent DRL algorithms and their use in various fields were surveyed by Gronauer and Diepold (2022), Nguyen et al. (2020) and Du and Ding (2020). These survey articles provide a detailed discussion of various challenges and solutions in the field of multi-agent DRL. Pateria et al. (2021) conducted a survey of the various Hierarchical Reinforcement Learning (HRL) methodologies in order to address the difficulties of using HRL to learn subtask discovery, multi-agent learning, hierarchical policies and transfer learning. The interconnections and differences between the Inverse Optimal Control and Inverse Reinforcement Learning techniques were reviewed by Azar et al. (2020) to fill the research void in the current body of literature.

The presented survey in this paper provides an overall understanding of RL foundations, algorithms and promising research directions and is not confined to any specific application field. This survey article briefly discusses the fundamentals of RL and the significant pioneering work in the field. In this survey, the primary goal is to give new researchers comprehensive subject information about the RL approach. Nearly all model-free pioneering DRL algorithms are covered in detail, along with a brief overview of model-based DRL techniques.

1.2. Contributions of this survey

This survey aims to provide a thorough analysis of RL literature and different RL methodologies and a summary of recent notable work and some promising research fields. The main contributions are as follows:

- We conduct a comprehensive survey of the historical and recent work done so far in the field of RL.
- Along with a comprehensive analysis and comparison of model-free DRL algorithms, the survey gives a brief overview of the basic RL algorithms.
- A brief description of model-based RL algorithms and multi-agent RL algorithms is also included in the survey.
- This survey also reviews several recent RL efforts that applies modern DNN in policy and value networks or state feature extraction for a variety of application fields.
- In order to provide guidance for future studies on the scalability, sample efficiency, effectiveness, and theoretical robustness of RL, we outline a number of significant research fields.

The outline of the paper is as follows: Some basic concepts of RL are discussed in Section 2. Section 3 covers the model-free RL algorithms for discrete state and action spaces. Section 4 gives the idea of value function approximation in RL. Some function approximation based policy prediction and policy control algorithms are also discussed in this section. Section 5 covers some important model-free DRL algorithms and some model-based approaches. Section 6 describes the current research work and some important research fields in RL. Finally, Section 7 draws some conclusions.

2. Basics of reinforcement learning

In RL literature, the learner or decision-maker is referred to as the agent and the world in which the agent lives and interacts is referred to as the environment. The agent can interact with the environment by performing some actions, but those actions do not affect the environment's laws or dynamics. The present condition of the environment is known as the state in RL literature, and RL agents execute actions depending on the state and reward signals.

Rewards and penalties are used to train RL agents. When our agent makes a good decision, we give it a reward; when it makes a bad one, we give it a penalty. The agent policy is updated by RL algorithms based on prior rewards obtained through all possible actions in various states. RL's essential goal is to determine a sequential decision-making policy or strategy that leads us to maximum total rewards or returns. In most RL problems, this return is usually a user-defined reinforcement signal accumulating from immediate rewards. In this way, the agent tries to estimate an optimal policy with the help of a reward signal generated by agent environment interaction. Fig. 2 represents the basic framework as discussed in this section.

If the agent takes any action u in a given state x, the expected immediate reward can be expressed as $r_{k+1}(x,u) = E[r_{k+1}|x_k = x, u_k = u]$. If the sequence of immediate rewards received after time step k is $\{r_{k+1}, r_{k+2}, r_{k+3}\ldots, r_K\}$ for an episodic task, then the return G_k for this episodic task can be expressed as

$$G_k = r_{k+1} + r_{k+2} + r_{k+3} + \cdots + r_K \tag{1}$$

where K is the final time step. The agent receives some description of the state of the environment at each time step and selects an action based on that state and previously observed rewards for the same or similar states. The final state signal given to the RL agent is usually a preprocessed

Table 1
Summary of the influential work in the field of RL.

Reference	Topic	Details
(Thorndike, 1911)	Law of Effect	The "Law of Effect," which was inspired by the learning concept of trial-and-error learning, was first proposed by Edward Thorndike in 1911.
(Pavlov & Anrep, 1927)	The term reinforcement	In Pavlov's studies of conditioned reflexes, the word "reinforcement" was first used.
(Turing, 1948)	Pleasure-pain system	Alan Turing proposed a "pleasure-pain system" based on the Law of Effect in 1948.
(Shannon, 1950)	Computer program for playing chess	A computer program for playing chess was developed in 1950 by Claude Shannon.
(Shannon, 1952)	Theseus (maze running mechanical mouse)	Claude Shannon's work on a mechanical mouse that could navigate a maze in 1952 was one of the best examples of how trial-and-error learning may operate.
(Bellman, 1956)	Bellman equation	With the Bellman equation, Richard Bellman provided one of the optimal control problem solutions in 1956.
(Bellman, 1958)	Dynamic Programming (DP)	Richard Bellman developed the Dynamic Programming (DP) approach, which uses the Bellman equation to solve optimal control problems.
(Samuel, 1959)	Computer program for playing checkers	Arthur Samuel developed a computer program for game checkers using a TD-based learning approach in 1959.
(Howard, 1960)	Policy iteration approach	Ronald Howard introduced the policy iteration approach for MDPs in 1960.
(Klopf, 1972, 1975, 1982)	The revival of RL through trial-and-error learning	The area of RL was revived by Harry Klopf's work in the 1970s using trial-and-error learning.
(Witten, 1977)	TD(0)	Ian Witten made an important contribution to the field of TD learning in 1977 by introducing the concept of TD(0).
(Watkins, 1989)	Q learning	DP and online learning were combined by Chris Watkins in 1989 to develop Q learning.
(Tesauro, 1995)	Backgammon computer software	Backgammon computer software was developed by Gerald Tesauro in 1992 using TD learning.
(Sutton et al., 2000)	Policy Gradient method	An alternate method, introduced in 2000, updates the policy in accordance with the gradient of expected reward with respect to the policy parameters. The policy is explicitly represented by its own function approximator, independent of the value function.
(Campbell et al., 2002)	Deep Blue (by IBM)	The world chess champion Garry Kasparov was defeated by an IBM computer program named Deep Blue on May 11, 1997.
(Konda & Tsitsiklis, 2000, 2003)	Actor-critic algorithms	For the simulation-based optimization of a Markov decision process over a range of parameterized randomized policies, Konda and Tsitsiklis introduced and examined a class of actor-critic algorithms in 2000.
(Krizhevsky et al., 2012; Karpathy et al., 2014; Simonyan & Zisserman, 2015)	Rise of deep learning (2005–2015)	Computer vision, speech recognition, video games, board game software, natural language processing, and robotics are just a few of the domains where deep neural network architectures and various DL techniques successfully excelled (LeCun et al., 2015).
(Mnih et al., 2015)	DQN (by Google DeepMind)	A new DRL agent, a Deep Q-network, was developed leveraging recent developments in deep learning (Simonyan & Zisserman, 2015; Karpathy et al., 2014) that can learn effective policies directly from high-dimensional sensory inputs using end-to-end RL.
(Silver et al., 2016)	AlphaGo (by Google DeepMind)	In a full-sized game of Go, the new search algorithm Alpha Go defeated a human professional player by combining Monte Carlo simulation with policy and value networks.
(Silver et al., 2017a)	AlphaGo Zero (by Google DeepMind)	Without human input, direction, or domain knowledge outside of the game rules, AlphaGo Zero, a system built exclusively on reinforcement learning, outperformed AlphaGo and achieved superhuman performance.
(Silver et al., 2018)	AlphaZero (by Google DeepMind)	With only knowledge of the game's rules and no domain-specific expertise, the AlphaZero algorithm produced superhuman performance in numerous difficult board games such as chess, shogi and Go.
(Vinyals et al., 2019)	AlphaStar (by Google DeepMind)	AlphaStar is the first DRL agent to reach Grandmaster level in StarCraft II (a video game with multi-agent challenges) and the first to compete in the top league of professional players with the original version of the game.
(Berner et al., 2019)	Dota 2 (by OpenAI)	In the professional online game Dota 2, OpenAI Five became the first AI agent to defeat the world champions using an existing DRL technique.
(Schrittwieser et al., 2020)	MuZero (by Google DeepMind)	With no prior understanding of the underlying dynamics, the MuZero algorithm obtained superhuman performance in a variety of difficult and visually complex tasks by combining a learned model with a tree-based search.
(Jumper et al., 2021)	AlphaFold (by Google DeepMind)	AlphaFold is a pioneering computer technique that routinely predicts protein structures with an atomic level of precision even when no similar structure is available.
(Li et al., 2022)	AlphaCode (by Google DeepMind)	For programming problems that have never been seen before, a system called AlphaCode can develop a novel solution.

version of the original sensor data. The state can also have information about past observations, but the state signal should not contain all the information about the environment.

An ideal state signal that has only the present state information about the environment in a compact form is known as Markov. The conditional probability distribution of future states of a stochastic process has the Markov property if it depends only on the current state. Sequential decision-making problems can be efficiently modeled with the help of MDPs (Howard, 1960). RL represents the interaction between an agent and the environment in terms of states, actions, and rewards using the formal framework of MDP. A finite MDP can be defined entirely by all possible state and action sets with the one-step model dynamics of the environment. For a finite MDP, one-step model dynamics is defined by the state transition probability and expectation of reward for a given state-action pair (x, u) (Sutton & Barto, 2018),

$$p(x', r|x, u) = P\{x_k = x', r_k = r\,|x_{k-1} = x, u_{k-1} = u\} \quad (2)$$

Almost all RL algorithms use an estimated state value function $V_\pi(x)$ and the state action value function $Q_\pi(x, u)$ to determine how advantageous it is for the agent to be in a particular state or to perform an action in that state. The value functions are useful in RL for estimating optimal policies. These value functions are primarily based on expected future rewards or returns. The expected return for a given policy π is $V_\pi(x)$ if the agent starts from state x and then follows the policy π. For a given policy, the state value function $V_\pi(x)$ can be expressed as (Sutton & Barto, 2018):

$$V_\pi(x) = E_\pi[G_k|x_k = x] = E_\pi\left\{\sum_{n=0}^{\infty}\gamma^n R_{k+n+1}|x_k = x\right\}, \forall x \in X \quad (3)$$

where γ is a constant, $0 \leq \gamma \leq 1$, called the discount factor. The discount factor governs how much significance an RL agent gives to rewards in the future when compared to the present reward. If this parameter is set to 0, the agent will be entirely myopic, only learning about actions that result in the best immediate reward.

If an agent starts from the state x and performs an action u and follows the policy π thereafter, then the expected return is known as the state action value function $Q_\pi(x,u)$ and expressed as (Sutton & Barto, 2018):

$$Q_\pi(x,u) = E_\pi[G_k|x_k = x, u_k = u] = E_\pi\left\{\sum_{n=0}^{\infty}\gamma^n R_{k+n+1}|x_k = x,\ u_k = u\right\},$$
$$\forall x \in X \text{ and } u \in U \quad (4)$$

The state value $V_\pi(x)$ of a state would theoretically represent the average reward obtained by the agent for visiting it a large number of times. Similarly, the agent's average reward for repeatedly performing a fixed action u in a given state x would be the state's action value $Q_\pi(x,u)$. Mathematically we can calculate the value functions using Eq. (5) and Eq. (6) (Sutton & Barto, 2018).

$$V_\pi(x) = \sum_u \pi(u|x)\sum_{x',r} p(x',r|x,u)[r + \gamma V_\pi(x')] \quad (5)$$

$$Q_\pi(x,u) = \sum_{x',r} p(x',r|x,u)\left[r + \gamma\sum_u \pi(u'|x')Q_\pi(x',u')\right] \quad (6)$$

$V_\pi(x)$ and $Q_\pi(x,u)$ can be efficiently estimated using the agent's interaction data. The value functions expressed in Eq. (5) and Eq. (6) are known as Bellman equations, which represent a recursive relationship between the value functions of current and successor states. For finite MDPs, with the help of optimal value functions $V^*(x)$ and $Q^*(x,u)$, we can estimate an optimal policy π^* (Sutton & Barto, 2018).

$$V^*(x) = \max_\pi V_\pi(x) = \max_{u\in U}\sum_{x',r} p(x',r|x,u)[r + \gamma V^*(x')] \quad (7)$$

$$Q^*(x,u) = \max_\pi Q_\pi(x,u) = \sum_{x',r} p(x',r|x,u)\left[r + \gamma\max_{u'} Q^*(x',u')\right] \quad (8)$$

$$\pi^*(x) = \operatorname{argmax}_u Q^*(x,u) = \operatorname{argmax}_u \sum_{x',r} p(x',r|x,u)[r + Q^*(x',u')] \quad (9)$$

2.1. Model-based RL: Dynamic programming (DP)

Any model that an agent can use to estimate how the environment will react to its actions is referred to as a model of the environment. The central objective of model-based RL techniques is the planning of optimal policy using an environmental model that can predict the next state and reward given the present state and action.

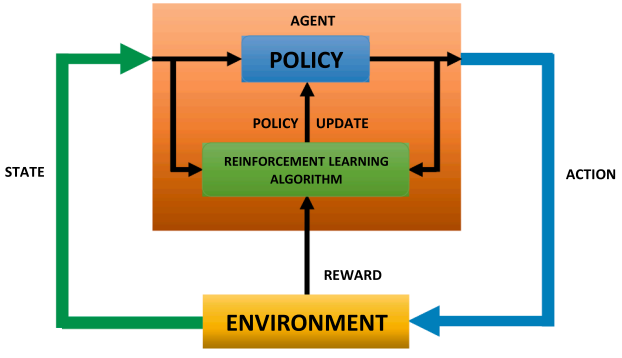

Fig. 2. Block diagram of reinforcement learning (reproduced from (MathWorks, 2023)).

DP is an algorithm that can be used to compute the agent's optimal policy if the complete dynamics or the model of the environment is given in the MDP formulation. The classical DP is not practically useful for large RL problems with continuous state and action spaces because it requires large computational power and complete environment dynamics. However, understanding DP is necessary to understand other RL approaches.

The whole idea of DP is based on Bellman equations. In DP literature, there are two main processes in the policy estimation of the agent. The first is policy evaluation, and the second is policy improvement. With the help of policy evaluation, we estimate the state values $V_\pi(x)$ for a given policy π. The complete policy evaluation process is given in Algorithm 1 (Sutton & Barto, 2018).

Algorithm 1: Policy evaluation for state value estimation

Input: Policy π, discount factor γ and a small constant $\sigma > 0$ for estimation accuracy
Initialize: $V(x)$ randomly $\forall\, x \in X$, except terminal states
repeat
  for all $x \in X$ **do**
    $V \leftarrow V(x)$
    $V(x) \leftarrow \sum_u \pi(u|x)\sum_{x',r} p(x',r|x,u)[r + \gamma V_\pi(x')]$
    $\beta \leftarrow \max_{x\in X}(|V - V(x)|)$
  end for
until $\beta < \sigma$
Output: $V_\pi(x) \approx V^*_\pi(x)$

This iterative evaluation procedure converges within a certain limit, but it must be stopped short of that range in practice. The algorithm evaluates the quantity $\max_{x\in X}|V_k(x) - V_{k+1}(x)|$ after each sweep over all states and ends when it is small enough.

With the help of the policy improvement method, we can improve the current agent policy $\pi(x)$ to $\pi'(x)$. Policy improvement refers to the process of creating a new policy $\pi'(x)$ that improves on an existing policy $\pi(x)$ by making it greedier with respect to the original policy's action value function.

$$\pi'(x) = \operatorname{argmax}_u Q(x,u) \quad (10)$$

By repeating the policy evaluation and policy improvement methods alternatively, we can reach to an optimal policy. The complete process of optimal policy estimation is known as policy iteration, which is presented in Algorithm 2 (Sutton & Barto, 2018).

Algorithm 2: Policy iteration for optimal policy estimation

Input: Discount factor γ and a small constant $\sigma > 0$ for estimation accuracy
Initialize: $V(x)$ randomly $\forall\, x \in X$, except terminal states
  : Random policy $\pi\ \forall x \in X$
Policy Evaluation
  repeat
    for all $x \in X$ **do**
      $V \leftarrow V(x)$
      $V(x) \leftarrow \sum_{x',r} p(x',r|x,u)[r + \gamma V_\pi(x')]$
      $\beta \leftarrow \max_{x\in X}(|V - V(x)|)$
    end for
  until $\beta < \sigma$
Policy Improvement
  for all $x \in X$ **do**
    $\pi(x) \leftarrow \pi$
    $\pi \leftarrow \operatorname{argmax}_{u\in U}\sum_{x',r} p(x',r|x,u)[r + \gamma V(x')]$
  end for $\pi = \pi(x)$; else go to *Policy Evaluation step*
Output: $\pi(x) \approx \pi^*(x)$

There is a drawback in policy iteration, which is each policy evaluation itself is an iterative process that requires multiple sweeps across all the

states. If we perform this complete policy evaluation, then the state values will take much time to converge. In the value iteration method, this policy evaluation process is truncated to reduce computation without losing the convergence guarantee by stopping the policy evaluation process just after one sweep through all states. So in value iteration (Sutton & Barto, 2018), we perform policy improvement with truncated policy evaluation steps.

3. Some model-free RL algorithms for discrete state and action spaces

The environment model consists of state transition probability and expectation of reward, but in any practical scenario, these may not be available for all possible states. Model-free RL methods use agent experience to learn the best possible value functions or policies directly without a complete environment model by estimating the optimal policy via a trial-and-error approach. The number of agent environment interaction data samples required for model-based algorithm training is lesser than that for model-free approaches. But still, model-based algorithms need model-free methods to construct the environment model. Model-free RL methods are advantageous for complex problems where building a sufficiently accurate environment model is challenging.

3.1. Monte Carlo (MC) learning

Monte Carlo (MC) is a model-free approach where an agent learns directly from experience data of episodic tasks. MC method requires a sequence of experience samples $(x, u, r, x^{'})$ generated in a simulated or real-world environment. A model is required for a simulated learning environment, which can generate the step reward and next possible state for the agent, but the policy and actions are not optimized based on the knowledge of the model. Basically, the MC method solves the RL problems by averaging the sample returns generated through agent-environment interactions. In the MC approach, the value and policy estimates are updated after the completion of an episode.

In MC policy evaluation methods, the agent tries to estimate value function $V(x)$ for a specified policy by averaging the sample returns received from the environment after each visit to the state x. After a sufficient number of visits, the state values converge to the expected values of states. Any given state x may be visited multiple times in an episode. In the first visit MC method, only the first visit to a given state is considered in an episode, whereas, for all visit MC methods, all the visits are considered. Algorithm 3 (Sutton & Barto, 2018) presents the first visit MC prediction method for a given policy.

Algorithm 3: MC policy evaluation for state value estimation

Input: Policy π and discount factor γ
Initialize: $V(x)$ randomly $\forall x \in X$, except terminal states
: Returns(x) ← an empty list, $\forall x \in X$
for all episodes **do**
  Generate an episode following policy π: $x_0, u_0, r_1, x_1, u_1, r_2 \cdots\cdots\cdots, x_{K-1}, u_{K-1}, r_K$
  $G \leftarrow 0$
  for all steps of the episode, $k = K-1, K-2, \cdots\cdots, 0$ **do**
    $G \leftarrow \gamma G_{k+1} + r_{k+1}$
    if x_k appears in $x_0, x_1, \cdots\cdots, x_{K-1}$ **then**
      Append G to Return(x_k)
    end if
  end for
  $V(x_k) \leftarrow$ average(Returns(x_k))
end for
Output: $V_\pi(x) \approx V_\pi^*(x)$

Here r is the step reward, and G is the return for each state. After each episode, return G is calculated backward from the terminal to the initial state.

For policy control, it is better to estimate action value functions. Consider a given environmental model M and random policy π, the agent will generate some learning data from different episodes in the form of $\{x_k^n, u_k, r_{k+1}^n, .., x_K^n\}_{n=1}^N$ M where N is the total number of episodes. For this model M, transition probabilities and expectation of immediate rewards are unknown. We can estimate the Q function by averaging the returns $Q(x_k, u_k) = \frac{1}{N}\sum_{n=1}^{N} G_k$ similar to policy evaluation. For a non-stationary environment, we can use an incremental update instead of an averaging approach: $Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha[G_k - Q(x_k, u_k)]$ where G_k is the total return, and α is the learning rate.

Our main aim here is to approximate optimal policies using MC updates. By estimating state action value functions, we can estimate the optimal policy. To identify an optimal strategy in RL, sophisticated strategies for balancing exploitation and exploration are required. Exploitation refers to taking advantage of present knowledge of the action values, while exploration implies attempting new actions in order to improve future action selection. Exploitation maximizes the expected step reward, but exploration may generate higher total rewards in the long run.

In Algorithm 4 for exploration, the ε-greedy policy is used, where ε is related to the exploration probability parameter. With this approach, the agent often exploits previously known information and explores new possibilities with some exploration probability. Mathematically ε-greedy policy can be represented as

$$\pi(u/x) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|u(x)| & \text{if } u = u^* \\ \varepsilon/|u(x)| & \text{if } u \neq u^* \end{cases}$$

Here $u \neq u^*$ means exploration and $|u(x)|$ is the number of possible actions in a given state. The probability of taking any random action u other than the current optimal action u^* is $\varepsilon/|u(x)|$ in a given state. The exploitation probability of taking the current optimal action u^* is $1 - \varepsilon + \varepsilon/|u(x)|$. Algorithm 4 (Sutton & Barto, 2018) presents the MC policy control or estimation method.

Algorithm 4: MC policy control for optimal policy estimation

Input: Discount factor γ and greedy policy parameter $\varepsilon > 0$
Initialize: ε-greedy policy
:$Q(x, u)$ randomly $\forall x \in X$ and $u \in U$, except terminal states
: Returns $(x, u) \leftarrow$ an empty list, $\forall x \in X, u \in U$
for all episodes **do**
  Generate an episode following policy π: $x_0, u_0, r_1, x_1, u_1, r_2 \cdots\cdots\cdots, x_{K-1}, u_{K-1}, r_K$
  $G \leftarrow 0$
  for all steps of the episode, $k = K-1, K-2, \cdots\cdots, 0$ **do**
    $G \leftarrow \gamma G_{k+1} + r_{k+1}$
    if the pair x_k, u_k appears in $x_0, u_0\ x_1, u_1 \cdots\cdots, x_{k-1}, u_{k-1}$ **then**
      Append G to Returns(x_k, u_k)
    end if
  end for
  $Q(x_k, u_k) \leftarrow$ average(Returns(x_k, u_k))
  $u^* \leftarrow \underset{u}{\text{argmax}} Q(x_k, u_k)$
end for
Output: $\pi(x) \approx \pi^*(x)$

MC methods have some major advantages over DP methods. One of the most important advantages is that MC is a model-free approach. The other advantage is that MC methods can be used with simulation or experience samples from the environment. In MC methods, the value function updates are only done for visited states, so the computational power requirement for MC methods is less than for DP methods.

3.2. Temporal difference (TD) learning

TD learning is the basic concept of most of the important RL algorithms. The idea of TD learning is the combination of DP and MC methods. It is a model-free approach like MC, and it learns with the help of previous value estimates like the DP method. In a simple MC update, we use a return G_k as a target for the update so that

$$V(x_k) \leftarrow V(x_k) + \alpha[G_k - V(x_k)] \tag{11}$$

In the MC method, values of each state visited in that episode are updated only after completion, whereas in TD(0) methods, a state value update is made after each state transition. A simple TD(0) update is shown in Eq. (12).

$$V(x_k) \leftarrow V(x_k) + \alpha[r_{k+1} + \gamma V(x_{k+1}) - V(x_k)] \tag{12}$$

Algorithm 5 (Sutton & Barto, 2018) summarizes the TD(0) policy evaluation procedure.

Algorithm 5: TD(0) policy evaluation

Input: Policy π, discount factor γ and learning rate α
Initialize: $V(x)$ randomly $\forall x \in X$, except terminal states
for all episodes **do**
  Initialize initial state x
  repeat for all visited states in an episode
    u ← Select an action using policy π
    Execute action u, observe reward r and next state x'
    $V(x) \leftarrow V(x) + \alpha[r + \gamma V(x') - V(x)]$
  until terminal state x
end for
Output: $V_\pi(x) \approx V^*_\pi(x)$

TD has two policy control methods: Sarsa on-policy and Q learning off-policy.

3.2.1. Sarsa

Sarsa is an on-policy TD method for policy control. Sarsa estimates the Q value functions with the help of TD updates to estimate the optimal policy. The Sarsa update is shown in Eq. (13) (Sutton & Barto, 2018).

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)] \tag{13}$$

Algorithm 6 (Sutton & Barto, 2018) presents the Sarsa on-policy TD control.

Algorithm 6: Sarsa

Input: Arbitrary policy π, discount factor γ and learning rate α
Initialize: $Q(x, u)$ randomly $\forall x \in X$, $u \in U$, except terminal states
for all episodes **do**
  Initialize initial state x
  repeat for all visited states in an episode
    u ← Select an action using arbitrary policy π
    Execute action u, observe reward r and next state x'
‘7: $Q(x, u) \leftarrow Q(x, u) + \alpha[r + \gamma Q(x', u') - Q(x, u)]$
  until terminal state x
end for
Output: $\pi(x) \approx \pi^*(x)$

3.2.2. Q-learning

In Q learning, the updation of the learned $Q(x, u)$ function is done with the help of the optimal Q function of the next state, as shown in Eq. (14) (Sutton & Barto, 2018). When compared to Sarsa, Q learning helps in early convergence.

$$Q(x_k, u_k) \leftarrow Q(x_k, u_k) + \alpha\left[r_{k+1} + \gamma \max_u Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)\right] \tag{14}$$

So, in Algorithm 6, line no.7, we can simply substitute the update Eq. (13) with Eq. (14) for Q learning off-policy TD policy control algorithm. The rest of the steps will remain the same.

3.3. Expected Sarsa

Another model-free on-policy TD control method is Expected Sarsa (Van Seijen et al., 2009), a variant of the Sarsa method. Expected Sarsa uses information about stochasticity in the behavioral policy to reduce the variance in the updates. Here Q value updates are performed on the expected value $\mathbb{E}\{Q(x', u')\}$ rather than $Q(x', u')$, as was the case in Sarsa. With this setting, the update equation of Expected Sarsa is represented as

$$Q(x, u) \leftarrow Q(x, u) + \alpha[r + \gamma V(x') - Q(x, u)] \tag{15}$$

where $V(x') = \sum_u \pi(u|x') Q(x', u)$. So, in Algorithm 6, line no.7, we can simply substitute the update Eq. (13) with Eq. (15) for the Expected Sarsa algorithm. The rest of the steps will remain the same. Zero variance is observed in Expected Sarsa updates for the deterministic environment. Expected Sarsa provides faster learning and experiments on multiple problems confirm that this method has major advantages over Sarsa and Q learning methods.

3.4. Double Q-learning

Sometimes, due to large overestimations, Q-learning performs poorly in stochastic environments. We use the maximum Q value approximation in Q learning for the expected maximum Q value, producing a positive bias. Double Q learning (Hasselt, 2010) solves this positive bias problem using a double estimator method for the approximation of expected maximum Q values. This new off-policy RL algorithm performs better in some of the tasks where Q learning failed miserably due to the overestimation issue. Maze problem and roulette game are some of the tasks where Double Q-learning quickly reached better performance levels. The procedure of Double Q learning is shown in Algorithm 7 (Hasselt, 2010).

Algorithm 7: Double Q-learning

Input: Arbitrary policy π, discount factor γ and learning rate α
Initialize: $Q^1(x, u)$ and $Q^2(x, u)$ randomly $\forall x \in X$, $u \in U$, except terminal states
repeat
  Select randomly from updating $Q^1(x, u)$ or $Q^2(x, u)$
  Select action u in current state x using arbitrary policy π and observe reward r and next state x'
  if updating $Q^1(x, u)$ **then**
    Estimate $u^* = \underset{u \in U}{\operatorname{argmax}}\, Q^1(x', u)$
    $Q^1(x, u) \leftarrow Q^1(x, u) + \alpha[r + \gamma Q^2(x', u^*) - Q^1(x, u)]$
  else update $Q^2(x, u)$
    Estimate $v^* = \underset{u \in U}{\operatorname{argmax}} Q^2(x', u)$
    $Q^2(x, u) \leftarrow Q^2(x, u) + \alpha[r + \gamma Q^1(x', v^*) - Q^2(x, u)]$
  end if
  $x \leftarrow x'$
until $Q(x, u) \approx Q^*(x, u)$
Output: $\pi(x) \approx \pi^*(x)$

4. Value function approximation (VFA) in RL

The value functions for all states are stored in a designated memory in methods like MC and TD. As we know, a state is an arrangement of observation features. A feature is a unique attribute or characteristic of a phenomenon that may be measured and a small change in the observation features results in a new state. So, for tabular methods like MC and TD, it is not practical to learn value functions for all the states if the

Fig. 3. Parameterized value function approximation.

problem's state and action spaces are too large. The memory and computational power requirement for such systems will be a huge challenge.

The Value Function Approximation (VFA) approach can be used for such problems. VFA brings the concept of generalization to the RL, where similar value estimates are assigned to states with similar state features. VFA is based on approximation, so these methods produce an approximation of optimal policy and values of the states. But still, this approximation approach produces faster computation due to the generalization of the states. Linear function approximators, ANN, Nearest Neighbor, and Decision Trees are some of the function approximation approaches. To optimize the results, it is better to use gradient-based optimization approaches. So linear function approximators and ANN function approximators are better choices.

In parameterized VFA, we represent the value functions $V_\pi(x)$ and $Q_\pi(x,u)$ with parameterized functions $\widehat{V}(x;\omega)$ and $\widehat{Q}(x,u;\omega)$ as shown in Fig. 3. Here $\omega \in \mathbb{R}^d$ represents the weight vector.

Function approximation is based on the supervised machine learning method, artificial neural networks (ANN), curve fitting, image and pattern recognition.

4.1. Function approximation based on-policy prediction

Linear VFA is one of the easiest and effective techniques for estimating $V_\pi(x)$ from data using a known policy π. A weighted sum of a linear combination of input observation features is used as an approximate function in the linear VFA method. For each state $\times$, there is a feature vector $Y(x) = [y_1(x), y_2(x), \cdots\cdots\cdots y_n(x)]^T$. In linear VFA, we represent a value function $\widehat{V}(x;\omega)$ for a given policy with a weighted linear combination of state features as

$$\widehat{V}(x;\omega) = \sum_{j=1}^{n} \omega_j Y_j(x) = \omega^T Y(x) \tag{16}$$

By using suitable loss function and gradient-based optimization techniques, we can approximate the value function $V_\pi(x)$. The complete procedure for a gradient-based MC algorithm for VFA is given in Algorithm 8 (Sutton & Barto, 2018).

Algorithm 8: Gradient-based MC Algorithm for VFA

Input: Policy π, learning rate α, discount factor γ and a differentiable function $\widehat{V}(x;\omega)$
Initialize: Value function weights $\omega \in \mathbb{R}^d$ randomly
for all episodes **do**
  Generate an episode using policy π: $x_0, u_0, r_1, x_1, u_1, r_2 \cdots\cdots\cdots, x_{K-1}, u_{K-1}, r_K$
  for all steps in the episode, $k = K-1, K-2, \cdots\cdots, 0$ **do**
    $G_k \leftarrow \gamma G_{k+1} + r_{k+1}$
    $\omega \leftarrow \omega + \alpha[(G_k - \widehat{V}(x_k;\omega)]\nabla_\omega \widehat{V}(x_k;\omega)$
  end for
end for
Output: $\widehat{V}(x;\omega) \approx V^*(x)$

Due to the biased targets, TD(0) VFA method does not produce the same convergence guarantee as gradient-based MC methods. Here the target estimate depends on the current value of the ω_k. It is not a true estimate, and we call it the semi-gradient method as we ignore the effect of ω on the biased target estimate. The complete procedure is given in Algorithm 9 (Sutton & Barto, 2018).

Algorithm 9: Semi-gradient TD(0) for VFA

Input: Policy π, learning rate α, discount factor γ and a differentiable function $\widehat{V}(x;\omega)$
Initialize: Value function weights $\omega \in \mathbb{R}^d$ randomly
for all episodes **do**
  Initialize x
  for all steps in the episode **do**
    Select action u following policy π
    Execute action u, observe reward r and next state x'
    $\omega \leftarrow \omega + \alpha[(r + \gamma\widehat{V}(x';\omega) - \widehat{V}(x;\omega)]\nabla_\omega \widehat{V}(x;\omega)$
    $x \leftarrow x'$
  end for
end for
Output: $\widehat{V}(x;\omega) \approx V^*(x)$

The convergence in semi-gradient approaches is not as robust as full gradient approaches. However, for linear cases, they produce reliable convergence. Continuous and online learning is also possible with semi-gradient methods. The other generalized form of VFA is state aggregation, where neighboring states are grouped, with one approximated value for each group (Singh et al., 1994).

Value function $\widehat{V}(x;\omega)$ can also be approximated using multi-layer ANN. A large number of complex functions can be approximated by ANN by adjusting the neuron's weights. There are some nonparametric function approximator based RL methods, which vary their number of parameters and shape as a function of the dataset, like kernel-based RL (Farahmand et al., 2008; Ormoneit & Sen, 2002), RL with Gaussian processes (Engel et al., 2005), memory-based function approximation (Atkeson et al., 1997), regression trees (Ernst et al., 2005), etc.

One of the important concepts of RL are the eligibility traces, which may be used to increase the convergence rate of VFA-based TD algorithms (Sutton & Barto, 2018; Singh & Sutton, 1996). Almost all TD methods, such as Sarsa and Q-learning, can be combined with this concept to create a more efficient learning method. The concepts of TD (0) and MC learning methods are combined and generalized in eligibility traces. With the help of eligibility traces, MC methods can be implemented successfully in online mode and for continuous state space problems.

Eligibility traces offer an elegant algorithmic mechanism with substantial computational advantages and allows consistent learning in time instead of being delayed until the end of the episode. Instead of the preceding N feature vectors, only a single tracing vector is required, known as eligibility trace or vector e_k. The traces are zero initially $e_k(x) = 0\,\forall\, x \in X$, and for each state visit, a short-term trace vector is initiated. This trace vector decays steadily over time. These eligibility traces mark the eligible states for learning. In accumulating traces, for each state visit, the update is done as (Sutton & Barto, 2018):

$$e_{k+1}(x) = \begin{cases} \gamma\lambda e_k(x) & \text{if } x \neq x_k \\ \gamma\lambda e_k(x) + 1 & \text{if } x = x_k \end{cases}$$

Here γ is a constant, $0 \leq \gamma \leq 1$, called the discount factor. The $\lambda \in [0,1]$ is a trace decay parameter. In a replacing trace (Singh & Sutton, 1996), for each state visit, the update is done as:

$$e_{k+1}(x) = \begin{cases} \gamma\lambda e_k(x) & \text{if } x \neq x_k \\ 1 & \text{if } x = x_k \end{cases}$$

Therefore, in accumulating trace, after each visit to a state, the eligibility trace builds up, whereas in replacing trace, the eligibility trace is reset to 1, irrespective of the prior trace value.

Similarly, for policy control, we have traces $e_k(x,u)$. In TD(λ) (Sutton & Barto, 2018; Van Seijen & Sutton, 2014) algorithms, the state value functions are estimated with the help of eligibility traces. The semi-gradient TD(λ) algorithm is shown in Algorithm 10 (Sutton & Barto, 2018).

Algorithm 10: TD(λ)

Input: Policy π, learning rate α, discount factor γ and a differentiable function $\widehat{V}(x;\omega)$
Initialize: Value function weights $\omega \in \mathbb{R}^d$ randomly
for all episodes **do**
  Initialize x
  e←0 (e is d-dimensional vector)
  for all steps in an episode **do**
    Select action u following policy π
    Execute action u, observe reward r and next state x'
    $e \leftarrow \gamma\lambda e + \nabla\widehat{V}(x;\omega)$
    $\delta \leftarrow r + \gamma\widehat{V}(x';\omega) - \widehat{V}(x;\omega)$
    $\omega \leftarrow \omega + \alpha\delta e$
    $x \leftarrow x'$
  end for
end for
Output: $\widehat{V}(x;\omega) \approx V^*(x)$

In the TD(λ) algorithm, we update the weights with the help of eligibility traces along with TD error. The linear TD(λ) algorithm showed good convergence properties (Dayan, 1992) for on-policy cases if the learning rate is reducing with time.

4.2. Function approximation based on-policy control

In this section, we will look at the on-policy control approaches using a parametric approximation of the action value function $\widehat{Q}(x,u;\omega) \approx Q^*(x,u)$, where ω is a weight vector. Semi-gradient Sarsa is a function approximation based on-policy control algorithm which is based on semi-gradient TD(0) algorithm. The weight update in one-step semi-gradient Sarsa algorithm for policy control is done as

$$\omega_{k+1} = \omega_k + \alpha[r + \gamma\widehat{Q}(x_{k+1},u_{k+1};\omega_k) - \widehat{Q}(x_k,u_k;\omega_k)]\nabla\widehat{Q}(x_k,u_k;\omega_k) \quad (17)$$

In policy evaluation, we estimate the value functions for a given policy by updating the weights. Policy improvement or control is achieved by combining this concept with greedy policy methods for better exploration.

4.2.1. Sarsa(λ)

For better learning speed, on-policy approximate control methods require efficient exploration. The concept of eligibility traces is also used with on-policy control methods to estimate the action value functions. When an element of ω_k contributes to the approximation of a value, then that specific element of eligibility trace e_k is bumped up and then starts decaying. Then learning starts with that element of ω_k if a nonzero TD error occurs before the trace is reduced to zero. For on-policy control, the true online TD(λ) method (Van Seijen & Sutton, 2014) can be easily updated. The algorithm is changed to Sarsa(λ) by simply using a state action value function approximation $\widehat{Q}(x,u;\omega)$ in place of a state value function approximation $\widehat{V}(x;\omega)$ (Rummery & Niranjan, 1994). The Sarsa (λ) algorithm is shown in Algorithm 11 (Sutton & Barto, 2018).

Algorithm 11: Sarsa(λ)

Input: Learning rate α, discount factor γ and a differentiable function $\widehat{Q}(x,u;\omega)$
Initialize: A differentiable function $\widehat{Q}(x;\omega)$ parameters ω_k and traces e_k
Observe initial state x_0, select random action u_0
for all steps $k = 0,1,2\cdots$ **do**
  Execute action u_k, observe next state x_{k+1} and reward r_{k+1}
  Select u_{k+1} with exploration policy based on $\widehat{Q}(x_{k+1},u_{k+1};\omega_k)$
  $\widehat{\delta}_k = r_{k+1} + \gamma\widehat{Q}(x_{k+1},u_{k+1};\omega_k) - \widehat{Q}(x_k,u_k;\omega_k)$
  $e_{k+1} = \gamma\lambda e_k + \nabla\widehat{Q}(x_k,u_k;\omega_k)$
  $\omega_{k+1} = \omega_k + \widehat{\delta}_k e_{k+1}$
end for
Output: $\widehat{Q}(x,u;\omega) \approx Q^*(x,u)$ and $\widehat{\pi}(x) \approx \pi^*(x)$

4.3. Function approximation based off-policy control

Here, we will discuss some model-free off-policy function approximation based policy control algorithms.

4.3.1. Fitted Q iteration

The fitted Q iteration (Ernst et al., 2005) is one of the off-policy control algorithms which estimates an optimal policy with the help of an approximated function $\widehat{Q}(x_k,u_k;\omega)$. To approximate the function $\widehat{Q}(x_k,u_k;\omega)$, this algorithm uses the set of the tuple (x,u,r,x') collected from previous experiences.

At iteration k, when the function approximation parameters are ω_k, the algorithm computes the Bellman target $Q_{k+1,i} = r_i + \gamma\max_{u'}\widehat{Q}(x_i',u';\omega_k)$ for each minibatch sample $i = 1,\cdots,M$. Then, using the training data samples, least squares regression is used to determine the next parameters ω_{k+1}.

The number of training iterations should be defined to stop this training. The following equation gives an error constraint on sub-optimality in terms of the number of iterations $\|\widehat{Q}^{\widehat{\pi}_N^*} - Q^{\pi^*}\|_\infty \le \frac{2\zeta\gamma^N}{(1-\gamma)^2}$ (Ernst et al., 2005). The maximum number of training iterations N can be restricted for the desired accuracy level and a given value of parameter ζ. When the stopping conditions are met, the final approximated control policy is calculated as follows: $\widehat{\pi}_N^*(x) = \operatorname{argmax}_{u\in U}\widehat{Q}_N(x,u)$. The complete procedure is given in Algorithm 12 (Ernst et al., 2005).

Algorithm 12: Fitted Q-iteration

Input: Discount factor γ, training dataset $\mathscr{D}$ and a differentiable function $\widehat{Q}(x;\omega)$
Initialize: Weights $\omega \in \mathbb{R}^d$ randomly
for $k = 0,1,2,\cdots.N$ **do**
  $Q_{k+1,i} = r_i + \gamma\max_{u'}\widehat{Q}(x_i',u';\omega_k)$, for $i = 1,\cdots,M$
  $\omega_{k+1} = \operatorname{argmin}_\omega\sum_{i=1}^M[Q_{k+1,i} - \widehat{Q}(x_i,u_i;\omega)]^2$
end for
Output: $\widehat{Q}^*$, $\widehat{\pi}^*$

4.3.2. Least Squares policy iteration (LSPI)

A new model-free off-policy method for policy control problems in RL uses linear VFA and policy iteration. This new method, called Least Squares Policy Iteration (LSPI) (Lagoudakis & Parr, 2003), is inspired by the Least Squares Temporal Difference (LSTD) learning algorithm (Bradtke & Barto, 1996) for prediction problems. Compared to TD algorithms, LSTD uses the experience samples more effectively. LSTD was

limited to policy evaluation with the help of state value functions. But LSPI can be used for policy control problems as it learns Q-function. LSPI allows policy improvement with the help of the Q function and policy iteration method. Compared to Q learning methods, LSPI performed better in the bicycle balancing and riding task (Lagoudakis & Parr, 2003). The complete LSPI procedure is shown in Algorithm 13 (Lagoudakis & Parr, 2003).

Algorithm 13: LSPI

Input: Discount factor γ and training dataset $\mathscr{D}$
1: Initialize: Policy π_0
2: **repeat** for all iteration $k = 0, 1, 2, \cdots .N$
3: $\mathscr{A}_k = \sum_{i=1}^{M} \phi(x_i, u_i)\left[\phi^T(x_i, u_i) - \gamma\phi^T(x_i', \pi_k(x_i'))\right]$
4: $\hbar_k = \sum_{i=1}^{M} \phi(x_i, u_i) r_i$
5: Solve $\omega_k = \mathscr{A}_k^{-1}\hbar_k$ to get parameter ω_k
6: $\pi_{k+1}(x) = \underset{u}{\operatorname{argmax}}\, \widehat{Q}(x, u; \omega_k)$
7: **until** $\pi_k = \pi_{k+1}$
Output: $\pi(x) \approx \pi^*(x)$

Here state action feature vector is represented by $\phi(x, u)$ and a linear architecture is used to approximate the state-action value function: $\widehat{Q}(x, u; \omega) = \sum_{j=1}^{M} \phi_j(x, u)\omega_j = \phi(x, u)^T\omega$. The LSPI and fitted Q iteration algorithm have similar convergence properties. If the error between action value functions $\|\widehat{Q}^{\pi_N^*} - Q^{\pi^*}\|$ remains less than the upper bound for every iteration, then the policy can be converged such that $\|\widehat{Q}^{\pi_N^*} - Q^{\pi^*}\|_\infty \leq \frac{2\zeta\gamma^N}{(1-\gamma)^2}$ (Lagoudakis & Parr, 2003; Lazaric et al., 2012). Generally, when the LSPI algorithm converges, it requires fewer iterations than the fitted Q iteration algorithm.

On-policy algorithms converge faster than off-policy and have lesser variance. In off-policy control methods, the vulnerability of instability and divergence increases whenever we face a deadly triad which is the combination of off-policy training, bootstrapping and function approximation. With linear function approximators, online TD learning gives some convergence guarantee (Tsitsiklis & Van Roy, 1997). A study on the convergence properties of a few approximate Q learning variants has been presented in Melo et al. (2008), but most of these convergence properties are only restricted to linear value function approximation, which is obviously not reasonable.

5. Deep reinforcement learning

Linear VFA requires a carefully hand-designed input features set, and it seldom works well without the right set of features, which is not always easy to select. VFA also has issues like correlations between training samples and non-stationary targets. An alternative is to use a much richer function approximation class like DNN function approximation, where we directly use states without requiring an explicit specification of features. It can represent complicated nonlinear functions using different activation functions. ANN is one of the best choices for nonlinear function approximation (Bertsekas & Tsitsiklis, 1996; Haykin, 2008), and in the last few years, deep neural networks are becoming popular (Mnih et al., 2015).

The first major success of ANN function approximation in RL was the Backgammon computer program in 1992. Gerald Tesauro developed it at IBM Research Center. Backgammon is a thousand years old two-player board game that is played on a one-dimensional track. In the field of game playing, TD Gammon was a fascinating work. It was successful in combining the fields of ANN and RL. For nonlinear function approximation, it combined the TD method to temporal credit assignment with the help of multilayer perceptron architecture. This self-playing RL approach produced more exciting results than supervised learning for some problems. After this success, considerable interest was developed within the ML research community to utilize the fundamental principles of TD Gammon's self-learning process (Tesauro, 1995).

Although RL had several significant success stories in the past (Kohl & Stone, 2004; Ng et al., 2006; Singh et al., 2002; Tesauro, 1995), previous approaches were limited to fairly low-dimensional tasks with hand-engineered and low-dimensional features. The main reasons for these limitations were memory and computational complexity (Strehl et al., 2006). The rise of DL due to DNN with powerful representation learning and function approximation properties has solved some of the most complex problems.

The DL is quite successful for problems such as speech recognition, object detection, and natural language translation (Lecun et al., 2015). DNN can automatically extract the low-dimensional features from high-dimensional input data like audio and pictures. Deep reinforcement learning (DRL) is a powerful method to introduce efficient function approximation and enable RL to solve some of the most complex learning problems, such as playing video games directly from image pixels (Mnih et al., 2015).

DRL has accomplished superhuman performance in board games like Go, Chess, and Shogi (Silver et al., 2016; Silver et al., 2017a; Silver et al., 2018; Schrittwieser et al., 2020). Through a trial-and-error approach, DRL can produce efficient autonomous RL agents that can learn optimal policies for complex real-world problems. It can also be implemented in robots to estimate the optimal policies directly from image inputs from the environment.

In recent years, two extraordinary success stories have entirely revolutionized the field of DRL. The first was the superhuman level performance of the DQN algorithm on Atari games, where they used game images as input states (Mnih et al., 2015). Google DeepMind developed a game-playing computer program AlphaGo that can play the board game Go. In October 2015 and March 2016, this computer program defeated some top Go players like Lee Sedol (Silver et al., 2016). In 2017, AlphaGo Master, the successor of AlphaGo, beat the world champion Ke Jie in a three-game match. Using the MC tree search method, AlphaGo and its successor AlphaGo Master successfully found the optimal moves based on previous knowledge learned by DRL methods. AlphaGo Zero (Silver et al., 2017a) is an updated version of AlphaGo. Without any prior knowledge and only by playing with itself, AlphaGo Zero surpassed the AlphaGo program's strength in 3 days and achieved the learning level of AlphaGo Master in 21 days. AlphaZero is a generalized version of AlphaGo to play Chess, Shogi, and Go (Silver et al., 2018).

In this section, we will discuss some of the important DRL algorithms, such as Neural Fitted Q iteration (NFQ) (Riedmiller, 2005), Deep Q Learning (DQN) (Mnih et al., 2013; Mnih et al., 2015), Double DQN (Hasselt et al., 2016), Dueling DQN (Wang et al., 2016), Prioritized Replay DQN (Schaul et al., 2016), Rainbow (Hessel et al., 2018), Policy Gradient (Sutton et al., 2000), Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016), Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016), Actor-Critic with Experience Replay (ACER) (Wang et al., 2017), Generalized Advantage Estimation (GAE) (Schulman et al., 2016), Twin Delayed DDPG (TD3) (Fujimoto et al., 2018), Soft Actor-Critic (SAC) (Haarnoja et al., 2018a; Haarnoja et al., 2018b), Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), and Proximal Policy Optimization (PPO) (Schulman et al., 2017). It is difficult to illustrate a precise, all-encompassing categorization of algorithms in the modern RL space. The basic classification of model-free DRL algorithms based on value and policy-based approaches is shown in Fig. 4.

5.1. Value optimization methods

In value optimization-based approaches, a random value function is chosen first for each state, and then a new value function is estimated

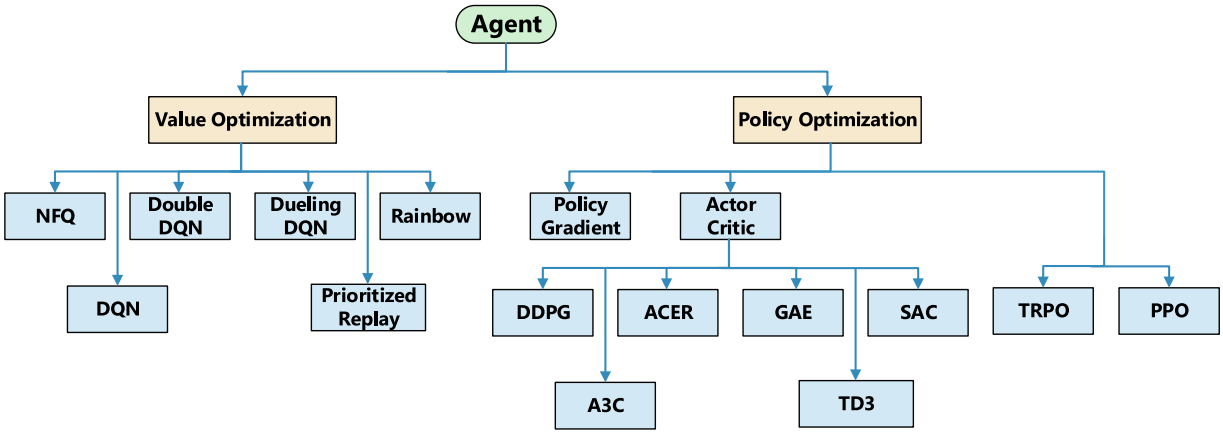

Fig. 4. Some important model-free DRL algorithms.

using learning data. This procedure is repeated until the optimal value function is estimated. The optimal policy is the one that follows the optimal value function and thereby gets updated implicitly. This subsection discusses some of the value optimization-based DRL approaches.

5.1.1. Neural fitted Q iteration (NFQ)

NFQ is a model-free approach for effective and efficient training of a Q value function approximated by ANN architecture. The offline updates were performed in the NFQ approach using experience samples instead of online updates (Riedmiller, 2005). The designed self-learning architecture successfully solved benchmark control problems from scratch, such as the double inverted pendulum with a specified control target (Riedmiller, 1999).

5.1.2. Deep Q learning (DQN)

The Q-Learning method creates a state action table for the working agent, which it can utilize to estimate the optimal policy. A better approach is required for problems having large state and action spaces. DQN is a DRL method that uses a DNN to approximate the Q value functions. The DQN algorithm was introduced by Google DeepMind in 2013 (Mnih et al., 2013) and later published with some improvements in 2015 (Mnih et al., 2015). After this work, a huge excitement existed in the field of AI because a DNN successfully learned control policies directly from high-dimensional sensory inputs like images using RL. In DQN, the states are fed into the DNN in the form of images, and it returns the estimated Q-values of all the permissible actions as output, with the help of which the optimal action can be selected very easily. The basic idea of DQN is shown in Fig. 5.

DeepMind used Arcade Learning Environment (ALE) to assess the performance level of the DQN algorithm (Bellemare et al., 2013). The ALE is an agent evaluation platform. Initially, ALE was designed for a classic game console, Atari 2600. Pong, Breakout, Space Invaders, and PacMan are some of the famous games of that game console. The DQN method was useful for problems with large state space and discrete and small action space. Atari games do not have high-dimensional action space and it is somewhere between 4 and 18, depending on the game. But state space is enormous because it is represented by pixels/images.

In 2013, DeepMind successfully demonstrated that with the help of a convolutional neural network (CNN), control policies could be learned directly from pixels data received from the environments (Mnih et al., 2013). In DQN, they used the basic idea of Q learning with gradient descent optimization methods. An experience memory (Lin, 1992) was also used in DQN to solve the data correlation problem. Experience or Replay memory is used in RL to store the agent's past experience samples (x_k, u_k, r_k, x_{k+1}) at each time step, which are pooled over multiple episodes.

Usually, a minibatch of experience samples is randomly sampled from this memory, which is then utilized for training RL agents. The reuse of data samples from the experience memory also improved data efficiency. Algorithm 14 (Mnih et al., 2013) represents the deep Q-

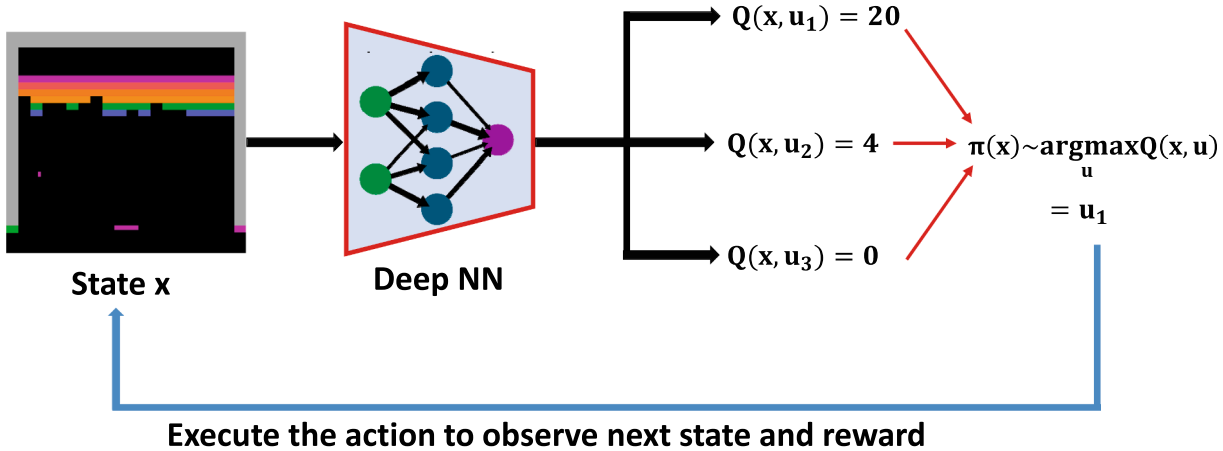

Fig. 5. The backup diagram of Deep Q Network (reproduced from (Amini & Soleimany, 2020)).

learning method.

Algorithm 14: DQN Algorithm

1: Initialize: Experience memory $\mathscr{D}$ and $\widehat{Q}$ values with randomly selected weights ω
2: **for** all episodes $n = 1, 2 \cdots\cdots N$ **do**
3: Initialize image sequence x_1 and preprocessed image sequence $\phi_1 = \phi(x_1)$
4: **for** all steps in each episode $k = 1, 2 \cdots\cdots K$ **do**
5: With probability ε select a random action u_k
6: Otherwise select $u_k = \underset{u}{\operatorname{argmax}} \widehat{Q}(\phi(x_k), u; \omega)$
7: Take action u_k in the simulator and observe the reward r_k and image x_{k+1}
8: Set x_{k+1} and preprocess $\phi_{k+1} = \phi(x_{k+1})$
9: Store transition $(\phi_k, u_k, r_k, \phi_{k+1})$ in memory
10: Sample random minibatch of transitions $(\phi_i, u_i, r_i, \phi_{i+1})$ from memory
11: Set $q_i = \begin{cases} r_i & \text{for terminal } \phi_{i+1} \\ r_i + \gamma \max\limits_{u'} \widehat{Q}(\phi_{i+1}, u'; \omega) & \text{for non} - \text{terminal } \phi_{i+1} \end{cases}$
12: Perform a gradient descent step on $(q_i - \widehat{Q}(\phi_i, u_i; \omega))^2$ with respect to the ω
13: **end for**
14: **end for**

For raw Atari frames, a preprocessed image sequence ϕ was used to reduce the input image dimensionality and computational demand. DQN outperformed the other RL algorithms (Bellemare et al., 2012; Hausknecht et al., 2014) on 6 out of 7 games and surpassed the expert human player's performance level on three of them.

In 2015, with an updated version of the DQN algorithm, DeepMind fully achieved professional human-level performance on 49 games (Mnih et al., 2015), using the same DNN architecture and hyperparameters. In the previous DQN method (Mnih et al., 2013), the same DNN was used to estimate the target q_i values and estimated $\widehat{Q}(\phi_i, u_i; \omega)$ values, but this approach created the problem of moving Q-targets. The first term of the TD error is the target value q_i, which is calculated as the immediate reward r_i plus the discounted maximum Q value for the succeeding state.

Here the weights ω were updated with the help of TD error, but the same DNN was used in both the target q_i and the $\widehat{Q}(\phi_i, u_i; \omega)$. This moved the estimated $\widehat{Q}(\phi_i, u_i; \omega)$ values closer to the target q_i by updated weights, but the updated weights also change the target q_i in the same direction. This gave rise to the situation of chasing a moving target, making the training difficult. To solve this moving Q target problem, DeepMind used a separate DNN for the target q_i values and fixed the weights ω^- of target DNN, replacing them with the weights ω of primary DNN only after every C time steps. They solved the moving target problem by substituting line 11 in Algorithm 14 (Mnih et al., 2013) with Eq. (18).

$$q_i = \begin{cases} r_i & \text{for terminal } \phi_{i+1} \\ r_i + \gamma \max\limits_{u'} Q(\phi_{i+1}, u'; \omega^-) & \text{for non} - \text{terminal } \phi_{i+1} \end{cases} \tag{18}$$

5.1.3. Double DQN

The Q-learning algorithm had an issue with the overestimation of Q values under some conditions. DQN algorithm, which was the combination of Q learning and DNN, faced this overestimation problem in some games while playing Atari 2600. The basic idea of Double Q learning (Hasselt, 2010), first introduced for tabular settings, was combined with a large-scale function approximation to avoid the overestimation problem in DQN. By dissolving the max operation in the target Q value estimation into action evaluation and action selection, Double DQN aims to reduce overestimations. The weights of the primary network (ω) were used to evaluate the greedy policy, whereas the weights of the target network (ω^-) are used to estimate the target value.

The Double DQN architecture proposed by Hasselt et al. (2016) successfully reduced the overestimations and performed much better on several Atari 2600 games than the DQN algorithm (Mnih et al., 2015). Algorithm 15 (Hasselt et al., 2016) presents the Double DQN method.

Algorithm 15: Double DQN Algorithm

1: Initialize: $\widehat{Q}(x, u; \omega)$ value with randomly selected weights ω
: Target $\widehat{Q}(x, u; \omega^-)$ with weights $\omega^- \leftarrow \omega$ and experience memory $\mathscr{D}$
2: **for** all episodes $n = 1, 2 \cdots\cdots N$ **do**
3: Observe state x_k
4: **for** all steps in each episode $k = 1, 2 \cdots\cdots K$ **do**
5: Select $u_k = \underset{u}{\operatorname{argmax}} \widehat{Q}(x_k, u; \omega)$
6: Take action u_k in the environment and observe the reward r_k and next state x_{k+1}
7: Store transition (x_k, u_k, r_k, x_{k+1}) in memory
8: Sample random minibatch of transitions (x_i, u_i, r_i, x_{i+1}) from memory
9: Set $q_i = \begin{cases} r_i & \text{for terminal } x_{i+1} \\ r_i + \gamma\, \widehat{Q}\left(x_{i+1}, \underset{u}{\operatorname{argmax}} \widehat{Q}(x_{i+1}, u; \omega); \omega^-\right) & \text{for non} - \text{terminal } x_{i+1} \end{cases}$
10: Perform a gradient descent step on $(q_i - \widehat{Q}(x_i, u_i; \omega))^2$ with respect to the ω
11: Update target network: $\omega^- \leftarrow \omega$
12: **end for**
13: **end for**

5.1.4. Dueling DQN

A new ANN architecture was introduced for several similar valued actions for improved policy evaluation. This new dueling NN design used separate estimators for state-dependent action advantage and state value functions. This dueling network architecture enabled the RL agent to outperform the other methods on Atari 2600 domain (Wang et al., 2016).

5.1.5. Prioritized replay DQN

In previous approaches, experience samples were sampled uniformly from the experience replay memory regardless of the importance of the samples. A prioritizing experience framework was developed for efficient learning to reuse significant data samples more frequently (Schaul et al., 2016). DQN with prioritized experience replay outperformed DQN (Mnih et al., 2015) on 41 out of 49 Atari games.

5.1.6. Rainbow

In the DRL community, numerous independent enhancements have been proposed for the DQN technique (Mnih et al., 2015). But, it was not clear which of these different improved approaches were complementary and could be combined effectively. Hessel et al. (2018) examined six variants of the DQN method and studied their different combinations. The study showed that the different combinations of DQN variants performed better than other methods on the Atari games and offered better data efficiency. They successfully combined several DQN enhancements into a single learning algorithm known as Rainbow (Hessel et al., 2018), which achieved state-of-the-art performance.

5.2. Policy optimization methods

All RL approaches covered in the preceding sections are value optimization methods, which use value functions to estimate the optimal policy. The policy optimization methods directly estimate the optimal behavior policy without estimating the value functions (Sutton et al., 2000). The value functions may still be utilized to optimize the behavioral policy weights parameters, but not for the selection of different actions. Policy-based RL methods are more valuable for continuous

space problems than value function-based methods, since the latter become computationally expensive due to infinite states and (or) actions to estimate the Q values in continuous space problems. Some important policy optimization methods are discussed in this section.

5.2.1. Policy gradient methods

In policy gradient (PG) approaches, the policy is modeled using a parameterized function $\widehat{\pi}(u|x;\Theta) = \mathscr{P}\{u_k = u|x_k = x, \Theta_k = \Theta\}$ which is the action selection probability at time k for a given state x with policy parameter Θ. Some scalar performance measure $J(\Theta)$ is used to estimate the optimal policy parameters Θ. The value of the start state is used as the performance measure for episodic tasks, and the average reward rate is used as the performance measure for continuous tasks.

In PG methods, the action selection probabilities $\widehat{\pi}(u|x;\Theta)$ change smoothly with policy parameter Θ, whereas in the greedy approach, the action selection probabilities may change drastically for a small change in the estimated values. So convergence guarantees for PG methods are better than value function-based methods. These policy parameters can be updated as:

$$\Theta_{k+1} = \Theta_k + \alpha\nabla_\Theta J(\Theta_k) \tag{19}$$

where $\nabla_\Theta J(\Theta_k)$ is the gradient of the performance measure. In PG approaches, the agent policy should be parameterized in such a way that the policy should be differentiable with respect to its parameters and has finite value for all $x \in X,\ u \in U$. To ensure exploration in PG methods, we usually make the policy stochastic $\widehat{\pi}(u|x;\Theta) \in (0,1)$ for all u and Θ. Stochastic policies are preferable for problems like card games, where they produce better results than deterministic policies.

The performance measure depends on the probabilities of action selection and the state distributions in which the action selections are completed, and both depend on the policy parameters, making the parameter estimation difficult. For a given state, the effect of the policy parameters on the parameterized policy can be computed, but the effect of policy parameters on the state distributions cannot be computed directly because these distributions are typically unknown and depend on the environment. This makes it impossible to estimate the gradient of the performance measure with respect to the policy parameters.

The PG theorem offers an analytical approach for the performance gradient with respect to the policy parameters, which does not include the gradient of state distribution. This analytical expression for episodic tasks is represented as (Sutton & Barto, 2018):

$$\nabla J(\Theta) \propto \sum_{x\in X} d_\pi(x) \sum_{u\in u} Q_\pi(x,u)\nabla\widehat{\pi}(u|x;\Theta) \tag{20}$$

where $d_\pi(x) = \lim_{k\to\infty} P(x_k = x|x_0, \pi_\Theta)$ is the stationary state distribution, which gives the probability that $x_k = x$ when starting from x_0 and following policy π_Θ for k time steps.

Based on the classical REINFORCE algorithm (Willia, 1992), the performance gradient can be expressed as (Sutton & Barto, 2018):

$$\nabla_\Theta J(\Theta) = \mathbb{E}_\pi\left[G_k \frac{\nabla_\Theta\widehat{\pi}(u_k|x_k;\Theta)}{\widehat{\pi}(u_k|x_k;\Theta)}\right] \tag{21}$$

The expectation in the above expression after sampling at each time step gives the performance gradient. Using this, the stochastic gradient ascent algorithm produces the REINFORCE update:

$$\Theta_{k+1} = \Theta_k + \alpha G_k \frac{\nabla_\Theta\widehat{\pi}(u_k|x_k;\Theta_k)}{\widehat{\pi}(u_k|x_k;\Theta_k)} \tag{22}$$

Using the mathematical identity $\nabla \ln x = \frac{\nabla x}{x}$ the last REINFORCE update can be expressed as:

$$\Theta_{k+1} = \Theta_k + \alpha G_k \nabla_\Theta \ln\widehat{\pi}(u_k|x_k;\Theta_k) \tag{23}$$

MC policy gradient algorithm (Sutton & Barto, 2018) is based on this idea is given in Algorithm 16.

Algorithm 16: REINFORCE: MC PG Control for episodic task

Input: A differentiable parameterized policy $\widehat{\pi}(u|x;\Theta)$ and learning rate α

```
Initialize: Policy parameter Θ ∈ R^d
for all episodes do
    Generate an episode (x_0, u_0, r_1, ……, x_{K−1}, u_{K−1}, r_K) following policy π̂(u|x;Θ)
    for all steps in the episode k = 0, 1,……..,K-1 do
        G_k ← Σ_{n=k+1}^{K} γ^k r_n
        Θ ← Θ + αG_k ∇_Θ ln π̂(u_k|x_k;Θ)
    end for
end for
```

REINFORCE algorithm has good theoretical convergence properties. However, due to the MC based approach, REINFORCE may have slow learning due to high variance (Willia, 1992; Baxter & Bartlett, 2001). The PG theorem Eq. (20) can be combined with arbitrary baseline b(x) as in Eq. (24) to reduce the variance.

$$\nabla J(\Theta) \propto \sum_{x\in X} d_\pi(x) \sum_{u\in U} (Q_\pi(x,u) - b(x))\nabla\widehat{\pi}(u|x;\Theta) \tag{24}$$

5.2.2. Actor-Critic method

In actor-critic methods, both value functions and policy approximations are made, where the learned approximate policy is referred to as 'actor' and the learned approximate value function is referred to as 'critic'. The basic actor-critic architecture is shown in Fig. 6.

Using value function approximation, these methods can tackle the high variance problem in policy gradient methods. This fundamental concept is given by policy gradient theorems (Crites & Barto, 1994; Konda & Tsitsiklis, 2003; Sutton et al., 2000). In the REINFORCE with baseline algorithm, the agent learns the state value function and policy, but this is not an actor-critic approach since the state value function is not used as a critic. REINFORCE with baseline algorithm learns slowly due to the MC approach and is not useful for continuous space problems.

TD methods can eliminate these problems, where a bootstrapping critic is used to solve the issues in PG methods. Based on TD methods, one-step actor-critic methods work in online and incremental modes, wherein a one-step return is used with a state value function baseline.

$$\omega_{k+1} = \omega_k + \alpha[r_{k+1} + \gamma\widehat{V}(x_{k+1};\omega) - b(x_k)]\frac{\nabla_\Theta\widehat{\pi}(u_k|x_k;\Theta_k)}{\widehat{\pi}(u_k|x_k;\Theta_k)}$$

$$= \omega_k + \alpha\delta_k\frac{\nabla_\Theta\widehat{\pi}(u_k|x_k;\Theta_k)}{\widehat{\pi}(u_k|x_k;\Theta_k)} \tag{25}$$

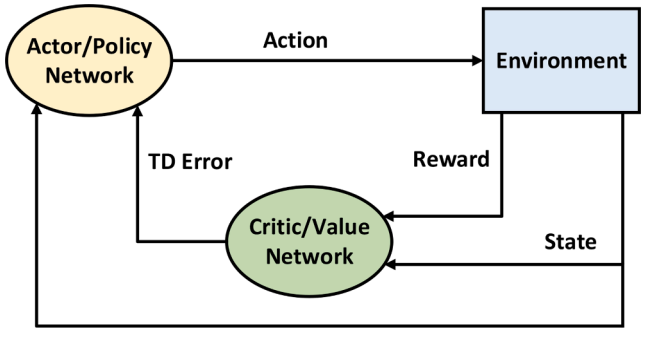

Fig. 6. Actor-Critic network (recreation based on (Sutton & Barto, 1998)).

The fully online incremental algorithm with semi-gradient TD(0) method is as follows (Sutton & Barto, 2018):

Algorithm 17: One-step Actor-Critic for optimal policy estimation

Input: A differentiable parameterized policy $\hat{\pi}(u|x;\Theta)$ and state value function $\hat{V}(x;\omega)$
: Learning rates $\alpha_\omega > 0$, $\alpha_\Theta > 0$
Initialize: Parameters $\Theta \in \mathbb{R}^d$ and $\omega \in \mathbb{R}^{d'}$ randomly
for all episodes **do**
  Initialize starting state of the episode
  for all steps in episode until x is not terminal state:
    Select action $u \sim \hat{\pi}(\bullet|x;\Theta)$
    Take action u and observe reward r and next state x'
    $\delta \leftarrow r + \gamma \hat{V}(x';\omega) - \hat{V}(x;\omega)$
    $\omega \leftarrow \omega + \alpha_\omega \delta \nabla_\omega \hat{V}(x;\omega)$
    $\Theta \leftarrow \Theta + \alpha_\Theta \delta \nabla_\Theta \ln \hat{\pi}(u|x;\Theta)$
    $x \leftarrow x'$
  end for
end for

A few landmark reference papers for actor-critic methods include Konda & Tsitsiklis (2000), where the first actor-critic configuration was presented, then the natural actor-critic method was presented in Peters and Schaal (2008a), and Bhatnagar et al., (2009), where they presented four new actor-critic based RL algorithms, with their convergence proofs.

5.2.2.1. Deep deterministic policy gradient (DDPG). DQN (Mnih et al., 2015) successfully solved the problems with continuous state and discrete action spaces. But generally, most of the real-world control tasks have continuous action spaces. DQN can not be employed directly in the continuous action domain since it depends on finding an action with a maximum Q value, which is impossible for continuous action space. The DDPG algorithm (Lillicrap et al., 2016) adapted the basic concept of DQN to the continuous action space with high-dimensional state spaces. DDPG is a model-free actor-critic approach that is based on a deterministic policy gradient method. With the same learning algorithm, hyperparameters, and DNN architecture, DDPG efficiently solved more than 20 complex control problems from high dimensional sensory input. The batch normalization technique (Ioffe & Szegedy, 2015), a development in deep learning, is used in DDPG along with the basic idea of DQN. The full concept of DDPG is given in Algorithm 18 (Lillicrap et al., 2016).

Algorithm 18: DDPG

Initialize: Critic $\hat{Q}(x,u;\omega_Q)$ and actor $\hat{\pi}(u|x;\Theta_\pi)$ with random weights ω_Q and Θ_π
: Target $\hat{Q}'$ and $\hat{\pi}'$ with weights $\omega_{Q'} \leftarrow \omega_Q$, $\Theta_{\pi'} \leftarrow \Theta_\pi$
: Replay memory $\mathscr{D}$
for all episodes $= 1, 2 \cdots\cdots N$ **do**
  Initialize a random process $\mathscr{N}$ for action exploration
  Receive initial state x_1
  for all steps in the episode $k = 1, 2 \cdots\cdots K$ **do**
    Select action $u_k = \hat{\pi}(u_k|x_k;\Theta_\pi) + \mathscr{N}_k$ according to the current policy and exploration noise
    Execute action u_k and observe the reward r_k and state x_{k+1}
    Store transition (x_k, u_k, r_k, x_{k+1}) in $\mathscr{D}$
    Sample a random minibatch of M transitions (x_i, u_i, r_i, x_{i+1}) from $\mathscr{D}$
    Set $y_i = r_i + \gamma \hat{Q}'(x_{i+1}, \hat{\pi}'(x_{i+1};\Theta_{\pi'});\omega_{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{M}\sum_i (y_i - \hat{Q}(x_i, u_i;\omega_Q))^2$
    Update the actor policy using the sampled policy gradient:
    $\nabla_{\Theta_\pi} J \approx \frac{1}{M}\sum_i \nabla_u \hat{Q}(x,u;\omega_Q)|_{x=x_i,\ u=\pi(x_i)} \nabla_{\Theta_\pi} \hat{\pi}(u|x;\Theta_\pi)|_{x_i}$
    Update target networks:
    $\omega_{Q'} \leftarrow \tau\omega_Q + (1-\tau)\omega_{Q'}$
    $\Theta_{\pi'} \leftarrow \tau\Theta_\pi + (1-\tau)\Theta_{\pi'}$
  end for
end for

5.2.2.2. Asynchronous advantage Actor-Critic (A3C). It was earlier assumed that the integration of DNN with online RL algorithms was necessarily unstable. But numerous methods have been introduced to solve this instability issue (Mnih et al., 2013, 2015; Riedmiller, 2005; Hasselt et al., 2016). Data correlation and nonstationary targets were some of the major reasons for these stability issues. To reduce the target nonstationarity and data correlations, researchers stored the agent's experience data in a buffer or replay memory, which could be batched (Hasselt et al., 2016) or sampled randomly (Mnih et al., 2013, 2015; Riedmiller, 2005). But this approach only restricts us to off-policy RL algorithms and requires large computational power and memory requirements. A3C algorithm (Mnih et al., 2016) provides a different approach for DRL.

Without replay memory, A3C used multiple agents simultaneously on different data samples asynchronously. This simple concept solved the data correlation problem. The computation power required was also much less, and A3C successfully solved complex problems with a single machine with multiple cores. A3C outperformed the prior approaches on the Atari gaming platform, and the training time with a multi-core CPU was just half of that with GPUs. The complete process for A3C is as follows (Mnih et al., 2016):

Algorithm 19: A3C

Initialize: Weight vectors $\Theta, \Theta', \omega_v$ and ω_v'
: Counters $K = 0$ and $k \leftarrow 1$
repeat
  Reset gradients: $d\Theta \leftarrow 0$ and $d\omega_v \leftarrow 0$
  Synchronize weights $\Theta' = \Theta$ and $\omega_v' = \omega_v$
  $k_{start} = k$
  Start from state x_k
  repeat
    Take action $u_k \sim \hat{\pi}(u_k|x_k;\Theta')$
    Observe reward r_k and the next state x_{k+1}
    $k \leftarrow k + 1$
    $K \leftarrow K + 1$
  until terminate x_k or $k - k_{start} == k_{max}$
  $G = \begin{cases} 0 & \text{for terminal } x_k \\ \hat{V}(x_k, \omega_v') & \text{for non terminal } x_k \end{cases}$
  for $i \in \{k-1, \cdots, k_{start}\}$ **do**
    $G \leftarrow r_i + \gamma G$
    $d\Theta \leftarrow d\Theta + \nabla_{\Theta'} \log\hat{\pi}(u_i|x_i;\Theta')(G - \hat{V}(x_i;\omega_v'))$
    $d\omega_v \leftarrow d\omega_v + \partial(G - \hat{V}(x_i;\omega_v'))^2/\partial\omega_v'$
  end for
  Update Θ using $d\Theta$ and of ω_v using $d\omega_v$ asynchronously
until $K > K_{max}$

Here Θ is the policy parameter,ω_v is the value function parameter, k is the step counter and K is the global counter for the updation of policy and value function parameters. This algorithm maintains a policy $\hat{\pi}(u_k|x_k;\Theta)$ and value function $\hat{V}(x_k,\omega_v)$, which are updated after some fixed steps or after reaching a terminal state.

5.2.2.3. Actor-Critic with experience replay (ACER). Photorealistic simulated training environments successfully trained the AI agents with remarkable cognitive abilities (Mnih et al., 2013, 2015, 2016; Schulman et al., 2016; Bellemare et al., 2013; Narasimhan et al., 2015; Brockman et al., 2016). But these richer realistic learning environments increased the cost of simulation.

Most of the PG methods were only applicable to continuous spaces or some specific problems. On-policy asynchronous advantage actor-critic (A3C) (Mnih et al., 2016), which was applicable to both continuous and discrete domains, was also not much sample efficient. ACER algorithm successfully solved some discrete and continuous space problems with remarkable sample efficiency and stability (Wang et al., 2017). A3C lays the foundation for ACER, although the former is an on-policy approach, whereas ACER is an off-policy approach. Managing the stability of the off-policy estimator is the main challenge in A3C. This issue was resolved by employing an effective TRPO, truncated importance weights with a bias correction trick, and a retrace estimator for Q-value. ACER's performance on Atari games was almost matched to DQN with a prioritized replay. The sample efficiency of the ACER approach was better

than A3C on discrete and continuous control problems, and the problem of data correlation was also reduced. The basic concept of ACER was based on variance reduction approaches, advances in DNN, the parallel training of RL agents (Mnih et al., 2016), and the Retrace algorithm (Munos et al., 2016). ACER method combined three innovative concepts: stochastic dueling networks, an efficient TRPO method (Schulman et al., 2015), and truncated importance sampling with bias correction.

5.2.2.4. Generalized advantage estimation (GAE). Generally, the main objective of RL is to maximize the total return for an agent policy. But the delayed reward produces a big challenge for effective learning. To solve this problem, value functions are used, which estimate the goodness of the action in a state before the occurrence of delayed reward. Generalized Advantage Estimation (GAE) (Schulman et al., 2016) algorithm optimizes the policy parameters using the value functions and offers a scheme for variance reduction in PG methods. The Trust Region Optimization technique was proposed for the value function, which is a robust and efficient ANN training method. Some extremely difficult 3D locomotion tasks were used to evaluate their method (Schulman et al., 2016). The results of these locomotion tasks were compared for different discount factor γ and trace decay parameter λ values. The best results were obtained for λ in the range [0.9, 0.99].

5.2.2.5. Twin delayed DDPG (TD3). The accumulation of error and overestimation bias for TD methods in actor-critic framework is studied in Fujimoto et al. (2018). TD learning approach has the problem of overestimation bias, which results in divergent behavior and suboptimal policy estimation. In continuous control tasks, deterministic policy gradient methods also suffer from this problem (Silver et al., 2014). The TD3 algorithm was proposed to address these issues, which successfully outperformed the other methods (Schulman et al., 2015, 2017; Brockman et al., 2016; Lillicrap et al., 2016). For variance reduction, a regularization strategy with a delaying policy was suggested in TD3 (Fujimoto et al., 2018). Algorithm 20 (Fujimoto et al., 2018) presents the TD3 method.

Algorithm 20: TD3

Initialize: Critics $\widehat{Q}_{\omega_1}$, $\widehat{Q}_{\omega_2}$, and actor π_Θ with random weights ω_1, ω_2, Θ
: Target $\omega_1{}' \leftarrow \omega_1$, $\omega_2{}' \leftarrow \omega_2$, $\Theta' \leftarrow \Theta$
: Replay memory $\mathscr{D}$
for k = 1, 2·····K **do**
  Choose action $u \sim \widehat{\pi}(u|x;\Theta) + \epsilon$ with exploration noise, $\epsilon \sim \mathscr{N}(0,\sigma)$
  Observe reward r and next state x'
  Store transition tuple (x, u, r, x') in $\mathscr{D}$
  Sample a random mini batch of M transitions (x, u, r, x') from $\mathscr{D}$
  $\tilde{u} \leftarrow \widehat{\pi}(u'|x';\Theta') + \epsilon$, $\epsilon \sim$ clip$(\mathscr{N}(0,\tilde{\sigma}), -c, c)$
  $y = r + \gamma \min_{i=1,2} \widehat{Q}(x', \tilde{u}; \omega_i')$
  Update critics $\omega_i \leftarrow \text{argmin}_{\omega_i} \frac{1}{M}\sum (y - \widehat{Q}(x, u; \omega_i))^2$
  if k mod delay time **then**
    Update Θ by the deterministic policy gradient:
    $\nabla_\Theta J(\Theta) = \frac{1}{M}\sum \nabla_u \widehat{Q}(x, u; \omega_1)|_{u=\widehat{\pi}(u|x;\Theta)} \nabla_\Theta \widehat{\pi}(u|x;\Theta)$
    Update target networks:
    $\omega_i' \leftarrow \tau\omega_i + (1-\tau)\omega_i'$
    $\Theta' \leftarrow \tau\Theta + (1-\tau)\Theta'$
  end if
end for

5.2.2.6. Soft actor-critic (SAC). Model-free DRL algorithms have been effective in numerous complex fields, like games (Mnih et al., 2013, 2016) and robotic control. But low sample efficiency in on-policy DRL methods such as TRPO (Schulman et al., 2015), A3C (Mnih et al., 2016), PPO (Schulman et al., 2017) and selection of hyperparameters are some of the major challenges for the implementation of DRL in real-world problems. Off-policy methods (Mnih et al., 2015) have better sample efficiency, but they have convergence and stability issues (Maei et al., 2009), and these methods performed poorly in the continuous domain. DDPG (Lillicrap et al., 2016) is an effective method for the continuous domain, but it is extremely fragile and sensitive to hyperparameters selection (Duan et al., 2016; Henderson et al., 2018).

The idea of a maximum entropy framework to design a stable and efficient model-free DRL algorithm for the continuous domain is exploited in Rawlik et al. (2012) and Fox et al. (2016). This adds an entropy maximization term to the traditional RL reward function (Haarnoja et al., 2017; Thanh et al., 2008). Using a temperature parameter, we can recover the original goal, which was to maximize total return (Haarnoja et al., 2017). This maximum entropy approach significantly improves robustness and exploration, as demonstrated in Haarnoja et al. (2017).

SAC (Haarnoja et al., 2018a; Haarnoja et al., 2018b) is a maximum entropy-based off-policy DRL algorithm, which provides better stability and sample efficiency. SAC outperformed off-policy DDPG in high-dimensional complex tasks, like Humanoid (Duan et al., 2016). SAC also solves the instability and complexity problems associated with soft Q-learning based maximum entropy algorithms (Haarnoja et al., 2017). Algorithm 21 (Haarnoja et al., 2018a) presents the SAC method.

Algorithm 21: SAC

Initialize: Weight vectors ω, ω', $\bar{\omega}, \Theta$ and replay memory $\mathscr{D}$
for all iterations **do**
  for all steps in the environment
    $u_k \sim \widehat{\pi}(u_k|x_k;\Theta)$
    $x_{k+1} \sim p(x_{k+1}|x_k, u_k)$
    $\mathscr{D} \leftarrow \mathscr{D} \cup \{(x_k, u_k, r(x_k, u_k), x_{k+1})\}$
  end for
  for all gradient steps **do**
    $\omega \leftarrow \omega - \lambda_V \widehat{\nabla}_\omega J_V(\omega)$
    $\bar{\omega}_j \leftarrow \bar{\omega}_j - \lambda_Q \widehat{\nabla}_{\bar{\omega}_j} J_Q(\bar{\omega}_j)$ for $j \in \{1, 2\}$
    $\Theta \leftarrow \Theta - \lambda_\pi \widehat{\nabla}_\Theta J_\pi(\Theta)$
    $\omega' \leftarrow \tau\omega - (1-\tau)\omega'$
  end for
end for

5.2.3. Trust region policy optimization (TRPO)

Policy iteration (Bertsekas, 2005), policy gradient (Peters & Schaal, 2008b), and gradient-free optimization (Szita & Lorincz, 2006) are different types of policy optimization algorithms. Due to the easy understanding and implementation, gradient-free optimization techniques like covariance matrix adaptation and cross-entropy were used for many problems. For function approximations, gradient-based optimization techniques were quite successful for supervised learning tasks in the continuous domain. These concepts in RL produced effective learning for complicated policies. By minimizing the specific objectives, better guarantees for policy improvement were achieved (Schulman et al., 2015).

By applying some assumptions to the theoretical algorithm, a new algorithm TRPO (Schulman et al., 2015) was proposed. In TRPO, the optimization of parameterized policy π_Θ was based on the following expectations (Schulman et al., 2015)

$$\underset{\Theta}{\text{maximize}}\, \mathbb{E}_{x\sim\, \rho\Theta_{\text{old}},\ u\sim\, \mu}\left[\frac{\widehat{\pi}(u|x;\Theta)}{\mu(u|x)}\right]\widehat{Q}(x, u;\Theta_{\text{old}}) \tag{26}$$

$$\text{subjected to } \mathbb{E}_{x\sim\, \rho\Theta_{\text{old}}}\left[D_{KL}(\widehat{\pi}(\bullet|x;\Theta_{\text{old}})\|\widehat{\pi}(\bullet|x;\Theta))\right] \le \vartheta$$

where ρ is visitation frequency, μ is sampling distribution, D_{KL} is KL divergence (Haarnoja et al., 2017) between old and new policy, and ϑ is a constant. Vine and single path were the two variants of this algorithm (Schulman et al., 2015). Nonlinear policies with thousands of parameters were optimized using these model-free policy search algorithms. In the experiments, TRPO methods successfully learned difficult policies for tasks like walking, hopping, swimming, and Atari games (Schulman et al., 2015).

5.2.4. Proximal policy optimization (PPO)

Based on ANN function approximations, we have several efficient RL methods such as DQN (Mnih et al., 2015), vanilla policy gradient (Mnih et al., 2016), (Schulman, 2016) and TRPO (Schulman et al., 2015). DQN was not very impressive with continuous control problems, and vanilla PG methods had poor robustness and data efficiency, whereas TRPO was relatively complicated. Using first-order optimization, the new Proximal Policy Optimization (PPO) method produced a reliable performance with better data efficiency (Schulman et al., 2017).

In PPO, a new surrogate objective with clipped probability ratios was proposed. For each policy update, multiple epochs of stochastic gradient ascent were used. This approach performs data sampling from the policy and optimization of policies alternatively. Based on the simulated experimental results, it was found that the performance with clipped probability ratios was better than other methods with different surrogate objectives. The PPO performed better than other algorithms on

Table 2
Summary of model-free DRL algorithms.

Sr. No.	Algorithms	On/Off-policy	Value/ Policy-based	Performance evaluation	Results compared with
1	DQN (Mnih et al., 2015)	Off-policy	Value-Based	• Tested on 49 Atari 2600 games. • Achieved more than 75% of the human score on 29 Atari games.	• Linear function approximator and human players
2	Double DQN (DDQN) (Hasselt et al., 2016)	Off-policy	Value-Based	• Tested on 57 Atari 2600 games. • Achieved higher median and mean scores across 49 Atari games than DQN.	• DQN and human players
3	Dueling DQN (Wang et al., 2016b)	Off-policy	Value-Based	• Tested on 57 Atari 2600 games. • Achieved higher median and mean scores across 57 Atari games than DQN and human player.	• DQN and human players
4	Prioritized Replay DQN (Schaul et al., 2016)	Off-policy	Value-Based	• Tested on 57 Atari 2600 games. • Achieved higher median and mean scores across 49 games than DQN. • Achieved higher median and mean scores across 57 Atari games than Double DQN.	• DQN and DDQN
5	Rainbow (Hessel et al., 2018)	Off-policy	Value-Based	• Tested on 57 Atari 2600 games. • Achieved higher median scores across all 57 Atari games than other approaches.	• DQN, DDQN, Prioritized DDQN, Dueling DDQN, A3C, Noisy DQN (Fortunato et al., 2017), Distributional DQN (Bellemare et al., 2017), and Rainbow
6	DDPG (Lillicrap et al., 2016)	Off-policy	Actor-Critic	• Tested on more than 20 simulated physics tasks in MuJoCo and TORCS (Wymann et al., 2013) environments. • Achieved higher scores across all tasks than Deterministic Policy Gradient (DPG) (Silver et al., 2014)	• DPG
7	A3C (Mnih et al., 2016)	On–policy	Actor-Critic	• Tested on 57 games of Atari 2600, TORCS 3D car racing simulator, continuous motor control problem in MuJoCo and 3D maze navigation problem in Labyrinth. • Achieved higher mean scores across all 57 Atari games than other approaches, median scores were also comparable. • Performed better on the TORCS simulator than a human tester. • Appreciable performance was found on MuJoCo and Labyrinth simulator.	• DQN, Gorilla (Nair et al., 2015), DDQN, Dueling DQN and Prioritized DQN
8	ACER (Wang et al., 2017)	Off-policy	Actor-Critic	• Tested on 57 Atari 2600 games and some continuous control tasks in MuJoCo. • Achieved higher median of the normalized human score across all 57 Atari games than DQN, Prioritized DDQN and A3C. • On MuJoCo continuous control problems, it performed better than A3C.	• DQN, Prioritized DDQN and A3C
9	GAE (Schulman et al., 2016)	Off-policy	Actor-Critic	• 3D locomotion tasks with simulated robots in MuJoCo.	–
10	TD3 (Fujimoto et al., 2018)	Off-policy	Actor-Critic	• Tested on continuous control tasks in MuJoCo. • Across all tasks, TD3 surpassed all other algorithms in terms of both performance and learning rate.	• DDPG, PPO, TRPO, SAC and ACKTR (Wu et al., 2017)
11	SAC (Haarnoja et al., 2018a)	Off-policy	Actor-Critic	• Tested on various challenging continuous space control problems from the OpenAI gym benchmark suite (Brockman et al., 2016) • Across all tasks, SAC outperformed all other algorithms.	• DDPG, PPO, SQL (Haarnoja et al., 2017) and TD3
12	TRPO (Schulman et al., 2015)	On–policy	Policy-Based	• Tested on Atari 2600 games and some continuous control tasks in MuJoCo. • Locomotion tasks are thought to be a difficult challenge, but the algorithm learned high-quality policy for them. • On Atari games, TRPO produced competitive results.	• Cross-Entropy Method (CEM) (Szita & Lorincz, 2006), Covariance Matrix Adaptation (CMA) (Hansen & Ostermeier, 1996) and Reward-Weighted Regression (RWR) (Peters & Schaal, 2007)
13	PPO (Schulman et al., 2017)	On–policy	Policy-Based	• Tested on simulated robotics tasks in OpenAI Gym environment, MuJoCo physics engine and Atari 2600 games. • Across all tasks, PPO matched or outperformed all other algorithms.	• A2C, CEM, Vanilla PG and TRPO

Table 3
Model-based DRL algorithms.

Algorithms	Tasks used for evaluation	Results compared with
Dyna (Sutton, 1990)	• (Theoretical Work)	(Theoretical Work)
Prioritized sweeping (Moore & Atkeson, 1993)	• Tested on various stochastic control problems. • Real-time control problems that are difficult to solve with other approaches were resolved successfully.	Temporal Difference
UCRL2 (Jaksch et al., 2010)	• (Theoretical Work)	(Theoretical Work)
PILCO (Deisenroth & Rasmussen, 2011)	• Tested on difficult, high-dimensional control problems (Cart pole and robotic uni-cycle). • Efficiently solved difficult control tasks.	–
Guided policy search (Levine & Abbeel, 2014)	• Tested on simulated robotic manipulation tasks. • Compared to earlier approaches, this method obtained better solutions with fewer samples.	iLQG (Li & Todorov, 2004), REPS (Peters et al., 2010), CEM, RWR (Peters & Schaal, 2007) and PILCO
Embed2Control (Watter et al., 2015)	• Tested on four visual control tasks. • Embed2Control attained performance similar to the real system model through optimal control.	Autoencoder (AE) and Variational Autoencoder (VAE)
Agrawal & Jia (Agrawal & Jia 2017)	• (Theoretical Work)	(Theoretical Work)
UCBVI (Azar et al., 2017)	• (Theoretical Work)	(Theoretical Work)
AlphaGo Zero (Silver et al., 2017a)	• Tested on board game Go. • AlphaGo Zero defeated the most powerful prior versions of AlphaGo.	AlphaGo Master and AlphaGo
VPN (Oh et al., 2017)	• Tested on 9 Atari 2600 games. • Outperformed DQN on 7 out of 9 Atari games.	DQN
MBMF (Nagabandi et al., 2017)	• Tested on MuJoCo locomotion tasks. • Achieved better results in very few steps than TRPO.	TRPO
I2A (Weber et al., 2017)	• Tested on Sokoban puzzle environment.	–
PETS (Chua et al., 2018)	• Tested on various tasks in the OpenAI Gym environment. • Significantly outperformed other approaches.	DDPG, SAC, PPO and Gaussian process dynamics models
AlphaZero (Silver et al., 2018)	• Tested on Go, Chess and Shogi. • Achieved superhuman performance in all of the games that were tested.	Stockfish (Stockfish, 2022), Elmo (Elmo, 2023), AlphaGo and AlphaGo Zero
MBVE (Feinberg et al., 2018)	• Tested on OpenAI Gym environment. • Outperformed other approaches on all tasks.	DDPG and IB (Kalweit & Boedecker, 2017)
UPN (Srinivas et al., 2018)	• Tested on simulated reacher robot. • Across all tasks, UPN performed better than other approaches and it was sample efficient than other approaches.	Auto-Regressive Imitation Learner and Reactive Imitation Learner
EULER (Zanette & Brunskill, 2019)	• (Theoretical Work)	(Theoretical Work)
MuZero (Schrittwieser et al., 2020)	• Tested on 57 Atari 2600 games and Go, Chess, Shogi board games. • Achieved better median and mean scores than other methods.	Ape-X (Horgan et al., 2018), R2D2 (Kapturowski et al., 2019), Impala (Espeholt et al., 2018), Rainbow, UNREAL (Jaderberg et al., 2016) and LASER (Schmitt et al., 2019)
SAVE (Hamrick et al., 2020)	• Tested on 14 Atari 2600 games. • Achieved better median and mean scores than other methods.	R2D2

continuous control tasks. Based on sample complexity PPO performed better than Advantage Actor-Critic (A2C) (Mnih et al., 2016) and similar to ACER on Atari games.

5.3. Summary of model-free DRL algorithms

Table 2 presents a summary of the model-free DRL algorithms covered in Sections 5.1 and 5.2.

In Table 2, algorithms from Sr. No. 1 to 5 are applicable to discrete (low-dimensional) action and continuous (high-dimensional) state space problems. While algorithms from Sr. No. 6 to 13 are applicable to continuous (high-dimensional) action and continuous (high-dimensional) state space problems.

Remark 1. *The performance summary given in Table 2 is obtained from the reference papers where the respective algorithms were originally proposed. These reference papers include a comparison of the performance of the proposed algorithm with other state-of-the-art algorithms. Most of these groundbreaking algorithms were evaluated on games from Atari 2600 and in environments such as MuJoCo and OpenAI Gym. Table 2 provides a brief summary of the performance of the algorithms on these test benchmarks.*

Only some model-free algorithms are covered in detail in this section, but model-based approaches are also extremely important in RL. Model-free RL involves learning directly from experience without explicit knowledge of the environment's dynamics. Model-based RL differs significantly from model-free RL in terms of its fundamental idea.

5.4. Model-based DRL algorithms

For optimization purposes, the sequential decision-making problems are generally modeled using MDPs. Combining RL with planning (Bertsekas, 2005) can solve these optimization problems, and this method is referred to as model-based RL (Deisenroth & Rasmussen, 2011). In this approach, the agent builds a model of the environment by learning the underlying dynamics of the system, such as how changes in the state of the system lead to changes in the rewards and transitions to new states. This model can predict the outcomes of different actions in different situations, which can guide the agent's decision-making process. The agent can also use this model to plan its activities to maximize the total reward. For example, the agent might use the model to simulate different actions and evaluate their expected outcomes, choosing the action that is most likely to lead to the highest reward. Model-based RL first needs the dynamics model to be approximated. Dealing with uncertainty resulting from sparse data, environment stochasticity, multi-step prediction, representation learning approaches, non-stationarity and partial observability for state and temporal abstraction are some

Table 4
Some recent notable DRL algorithms.

Algorithms	Details
MuZero (Schrittwieser et al., 2020)	• Developing intelligent agents with the ability to plan has been a major obstacle in AI. While tree-based planning approaches have proven successful in games such as Go and chess, where a proper simulator exists, real-world problems are often characterized by complex and unknown environmental dynamics. • Google DeepMind introduced the MuZero algorithm, which utilizes a combination of a learned model and a tree-based search to achieve exceptional performance in visually intricate and challenging tasks, with no prior information of the inherent dynamics of the system. • When assessed in 57 various Atari games, a widely used benchmark for testing AI methods where model-based planning techniques have traditionally underperformed, MuZero attained a new state of the art. • MuZero also matched AlphaZero's superhuman performance in chess, shogi and Go, despite lacking knowledge of the game rules.
Agent57 (Badia et al., 2020a)	• In Badia et al. (2020a), the authors present a RL agent called Agent57 that is able to outperform the Atari human benchmark on 57 Atari 2600 games. • This approach includes an adaptive mechanism to select which policy to focus on during training and a unique way of designing the ANN architecture that leads to more reliable and stable learning. • The agent achieved high scores on many of the games, surpassing the state-of-the-art performance of RL agents on these games.
Dreamer (Hafner et al., 2020)	• In order to make learning complex behaviors easier, learned world models review an agent's experience. There are numerous potential ways to derive behaviors from high-dimensional sensory inputs, even though building world models from them is becoming possible through deep learning. • Hafner et al. (2020) introduced Dreamer, a RL agent that only uses latent imagination to solve long-horizon problems from images. By propagating analytical gradients of learned state values back through imagined paths in the condensed state space of a learned world model, it effectively learns behaviors. • Dreamer outperformed existing methods in terms of data efficiency, calculation time, and end performance on 20 difficult visual control tests.
AlphaFold (Jumper et al., 2021)	• Proteins play a vital role in sustaining life, and comprehending their structure can help in knowing their function. Despite determining the structures of roughly 100,000 distinct proteins through a considerable experimental effort, this only accounts for a small percentage of the billions of well-known protein sequences. The "protein folding problem," or predicting a protein's three-dimensional structure purely from its amino acid sequence, has been a persistent scientific challenge for more than 50 years. • Despite the latest advances, current techniques still lack atomic precision, especially when a homologous structure is unavailable. Google DeepMind introduced AlphaFold, which can make an atomic precision prediction of protein structures even in the absence of a similar structure. • In the fourteenth Critical Assessment of Protein Structure Prediction (CASP14), AlphaFold was validated and showed accuracy that, in the majority of cases, is superior to other experimental structures.
AlphaCode (Li, et al., 2022)	• Despite the fact that recent large-scale language models have shown the ability to produce programming code, they still struggle when faced with more complex and unfamiliar problems that require advanced problem-solving skills, such as understanding algorithms and complex natural language. • To overcome this challenge, Google DeepMind introduced AlphaCode (Li et al., 2022), a code generation system that can generate original solutions for these complex problems. • AlphaCode regularly placed in the top 54.3% for competitions with more than 5,000 members in simulated evaluations of recent programming competitions on the Codeforces platform.
AlphaTensor (Fawzi et al., 2022)	• Improving the speed of fundamental computations is essential, as it can have a broad impact on many systems, including scientific computing and artificial neural networks. Matrix multiplication is a critical task that is ubiquitous in these systems. • To address this, Google DeepMind introduced AlphaTensor, a DRL approach based on AlphaZero. The agent is trained to discover tensor decompositions within a finite factor space for matrix multiplication. • For several matrix sizes, AlphaTensor developed algorithms that outperformed the state-of-the-art complexity, including 4×4 matrices, where it improved on Strassen's two-level algorithm, which had not been surpassed in the 50 years since its discovery.
DiL-piKL (Bakhtin et al., 2022)	• Self-play RL has been effective in purely adversarial games like Go, chess and poker, but it is insufficient for attaining the best results in fields that require human cooperation. • Bakhtin et al. (2022) propose the DiL-piKL planning algorithm as a solution to this problem. It regularizes a reward-maximizing policy in the direction of a human imitation-learned policy, which is a no-regret learning method under a modified utility function. • They train an agent named Diplodocus using RL-DiL-piKL, and evaluate it in a 200-game no-press diplomacy competition with 62 human competitors of various skill levels. According to an Elo rating model, the two Diplodocus agents come in first and third place and have greater average scores than any other player who played more than two games.
DreamerV3 (Hafner et al., 2023)	• The attainment of general intelligence necessitates the ability to complete tasks in various domains. RL algorithms possess this capability but are restricted by the sources and information needed to calibrate them for new problems. • To address this limitation, Google DeepMind introduced DreamerV3, a scalable and general algorithm grounded in world models that surpasses previous methods across various tasks with fixed hyperparameters. • It has been found that DreamerV3 has good scaling properties, and thus larger models perform better in terms of data efficiency and overall performance. As a difficult feat in AI, DreamerV3 is the first algorithm to acquire diamonds in Minecraft from the start without using any human data.

major issues in model learning. Aspects like targeted exploration, data efficiency, explainability, safety and transfer learning are the few advantages of model-based RL.

Model-based RL is a powerful approach that has been used for various tasks, including robot control, game playing, and natural language processing. It has the advantage of being able to handle complex environments and long-term planning, but it can be computationally intensive and may require a huge volume of data to learn an accurate model of the environment. However, model-based RL has had significant success in the past (Silver et al., 2017a; Silver et al., 2017b; Silver et al., 2018; Schrittwieser et al., 2020; Levine & Koltun, 2013; Deisenroth & Rasmussen, 2011). Moerland et al. (2022) and Luo et al. (2022a) thoroughly summarize the work in this field. Several significant model-based RL algorithms are listed in Table 3.

Both model-based and model-free approaches have their own benefits and drawbacks. Model-free approaches are generally easier to implement and can learn from experience more quickly, but they may be less efficient and may require more data to reach good performance. Model-based approaches can be more efficient and require fewer data, but they may be more difficult to implement and less robust to environmental changes.

5.5. Some recent notable DRL algorithms

Along with the model-free and model-based DRL algorithms covered in previous sections, some notable DRL algorithms have emerged in recent years. In this section, a summary of such algorithms is presented in Table 4.

6. Current research work in RL

Even though DRL has produced some fantastic results in various fields, it is difficult to replicate these results for problems in every field. Each learning problem has its own set of challenges and constraints. When using DRL on optimization problems, there are plenty of practical issues. Not every system will experience all of these obstacles, but in many situations, all of them will be present to some extent. Sample complexity, off-policy learning with limited data, high-dimensional continuous state and action spaces, partial observability, non-stationarity, observation delays, and multi-objective reward function design are some of the major practical issues with DRL.

Low sampling efficiency is one of the most significant challenges in DRL due to the need for continuous exploration. In addition to better optimization and data sampling techniques (Schulman et al., 2015; Sun et al., 2020b), better feature learning is a good way to deal with the problem of low sample efficiency. Despite the fact that DNN has delivered a high level of strength in representation learning, it still remains the key hurdle to further DRL algorithm progress due to the high dimensionality of the action and state spaces in many real-world problems. Various state representation learning schemes have been proposed to address this issue (Zhao et al., 2020).

DRL algorithms have been using Experience Replay (ER) (Lin, 1992) as a key component to improve sample efficiency. ER allows off-policy RL algorithms to update current policy by reusing previous experiences, which improves the sample efficiency (Sun et al., 2020b). Experience transitions are randomly replayed from the replay buffer in ER, ignoring the importance of various transitions. This is corrected in the Attentive Experience Replay (AER) (Sun et al., 2020b) algorithm which samples experience transitions based on how similar their states are to the agent's state, thereby enhancing the sample efficiency. In Pong et al. (2018), goal-oriented value functions are proposed. They improved the sample efficiency for a number of continuous control problems by proposing Temporal Difference Models (TDMs) that can be trained successfully by model-free methods and used for model-based control problems. The advantages of model-based and model-free RL methods are combined in these models.

Meta-RL aimed to address the sample efficiency problem by utilizing previous experience from a task to solve new tasks more rapidly. In Mendonca et al. (2019), the UC Berkeley research team showed that for multi-task demonstration learning data, this concept could be used for faster learning in RL. For Meta learning, they suggested an efficient and stable supervised imitation reinforcement learning method. Significant improvements in sample efficiency for different continuous control problems were demonstrated in this work.

Multi-tasking is one of the exciting fields in RL. With the same set of parameters, the DeepMind research team tried to solve a set of problems using a single RL agent (Espeholt et al., 2018). Their proposed Importance Weighted Actor-Learner Architecture (IMPALA) agent effectively used the resources in single-machine training. Without reducing the data efficiency, it also scaled to more machines. The success of this agent was demonstrated for multi-task RL on DMLab-30 and Atari-57 (Espeholt et al., 2018).

Deep RL algorithms also struggle to solve simple tasks in partial observability conditions where the complete state information of the system is not fully available. Due to this problem, RL agents generally face difficulties in 3D virtual learning environment. To overcome this problem, DeepMind introduced a new model MERLIN (Wayne et al., 2018) which successfully solved some benchmark problems from behavioral research in neuroscience and psychology.

There are various RL methods, each with a unique basic idea for generating learning data. In general, RL is difficult to apply to problems where data generation is dangerous, expensive, or time-consuming, and only a batch of interaction data is provided. In these circumstances, the offline RL (Fujimoto et al., 2019) approach can be utilized when the objective is to develop a highly profitable policy-based simply on a dataset of prior experiences with the environment. The term "offline" RL refers to a technique where the agent learns from data that has already been gathered rather than by interacting with the environment online. This strategy holds out the possibility of using such datasets to acquire policies without engaging in expensive or risky active exploration. However, the challenge is in the distributional difference between the offline dataset and the states the learnt policy visits. In model-based (Kidambi et al., 2020; Yu et al., 2020; Swazinna et al., 2021) and model-free (Kalyanakrishnan & Stone, 2007; Fujimoto et al., 2019) settings, offline RL has outperformed various state-of-the-art approaches. However, the fixed dataset being the only source of information for offline RL, this can limit the scope of what the agent can learn. It can be challenging for the agent to learn and adapt to changing settings or situations when using offline RL since real-time feedback is not received on the actions. Offline RL might not be as successful in dynamic contexts when the characteristics of the environment are continually changing.

In RL, the term "on-policy" refers to a learning approach where the agent's actions are based on the current policy being learned. This implies that the agent is following the current best estimate of the optimal policy as it learns, and updates to the policy are based on the actions taken by the agent. However, "off-policy" learning is a technique in which the agent's actions are dictated by a different policy from the one that is being learned. Here, the agent is employing a different "behavior" policy to investigate the environment and gain information rather than adhering to the current best estimate of the optimal policy. The target policy can then be updated using this data.

As the agent is not reusing the data for learning, on-policy methods may require a lot of interactions with the environment to produce a viable policy. On-policy RL approaches also have other significant problems, including high variance in estimates and sensitivity to the initial policy used. Because off-policy learning is based on experiences that were not produced by the agent's current policy, it can be less stable than on-policy learning. Off-policy learning can be prone to bias, especially if the behavior policy that was used to create the experiences differs greatly from the target policy. Nowadays, learning data is gathered and stored in an experience replay memory (Mnih et al., 2015) in

Table 5
Recent RL algorithms and other noticeable work.

Application area	Reference	Short description
Reinforcement learning	(Song et al., 2022)	A maximum entropy deep inverse RL algorithm (ME DIRL) based on the truncated gradient method and the Adaboost algorithm is proposed for better learning.
	(Dai et al., 2022)	For reward maximization, two novel actor-critic cooperative multi-agent RL algorithms are proposed.
	(Ladosz et al., 2022)	MOHQA, a modulated Hebbian network-based DQN algorithm, is introduced for solving the observable Markov decision process.
	(Liu & Feng, 2021)	The maximum mean difference, cross entropy approach in evolution policy and TD3 algorithm are coupled to provide a DEPRL algorithm based on diversity evolutionary policy for better cumulative return.
	(Hu et al., 2021)	A federated RL strategy is presented to enhance policy quality and training speed for each client, where reward shaping is used as federated information for independent clients with diverse assigned tasks.
	(Liu & Qian, 2021)	PG-Meta-MORL, a new prediction-guided multi-objective DRL algorithm based on *meta*-learning, is presented to handle potentially conflicting multi-objective optimization tasks with the effective adaptation of new objectives.
	(Rudin et al., 2021)	Demonstrated a complete GPU pipeline framework with parallel simulated robots training for challenging real-world control tasks using the DRL on-policy algorithm in a fairly short time
	(Jiang et al., 2021)	Two emphatic temporal difference DRL algorithms, WETD and NETD are introduced based on the emphatic temporal difference algorithm for improved convergence and stability.
	(Chen et al., 2021)	Introduced a decision transformer, an autoregressive model that portrays the RL problem as conditioned sequence modeling and predicts future actions depending on previous returns, states, and actions.
	(Kaiser et al., 2020)	Simulated Policy Learning (SimPLe) is introduced as a comprehensive model-based DRL algorithm that uses pixel observations and estimates efficient policies to play computer games in the Atari Learning Environment with minimal interactions with the environment.
	(Badia et al., 2020b)	Presented a RL agent capable of learning a variety of exploratory policies utilizing both dense and sparse reward possibilities.
	(Kirsch et al., 2020)	MetaGenRL, a novel off-policy meta general RL algorithm based on biological evolution, is proposed that uses a population of agents to meta learn generalized objective functions.
	(Fakoor et al., 2020)	Presented Meta-Q-Learning (MQL), an off-policy RL algorithm for *meta*-RL that showed comparable results to the state-of-the-art algorithms despite having a substantially easier training phase than previous algorithms.
	(Paine et al., 2020)	A framework for selecting hyperparameters for offline RL is proposed to select the best possible agent policy from a group of policies learned with different hyperparameters using given data.
	(Kumar et al., 2020)	To address the problem of overestimation of values in offline RL algorithms, a conservative Q-learning algorithmic framework (CQL) is proposed.
	(Lee et al., 2020)	For enhanced sample efficiency, the Stochastic Latent Actor-Critic (SLAC) model-free off-policy DRL algorithm is presented for learning from pixel inputs via representation learning.
	(Jaques et al., 2019)	Proposed a unified framework for efficient communication and coordination in Multi-agent RL by awarding agents for influencing the behaviors of other agents.
	(Rakelly et al., 2019)	PEARL, a novel off-policy meta RL algorithm, is proposed for enhanced sample efficiency with efficient exploration.
	(Zhou et al., 2019)	A hybrid Hierarchical Reinforcement Learning (hHRL) algorithm for multi-objective online navigation and guidance problems with a partially observable environment is proposed.
	(Krishnan et al., 2019)	SWIRL is a sequential windowed policy search inverse RL algorithm proposed in this paper, which is a combination of demonstration and exploration paradigms.
	(Ciosek et al., 2019)	Presented Optimistic Actor-Critical (OAC), a model-free DRL algorithm that improved sample efficiency in continuous control tasks by employing an enhanced exploration strategy.
	(Berner et al., 2019)	Developed a superhuman level self-playing DRL agent that defeated the world champion human Esport team in the Dota 2 computer game
	(Akkaya et al., 2019)	The OpenAI research team demonstrated that the AI robots trained in a simulation environment can solve the highly complex manipulation problem. With the help of their novel automatic domain randomization (ADR) algorithm, their AI robot successfully solved a Rubik's cube problem.
	(Riedmiller et al., 2018)	DeepMind introduced a novel RL approach called Scheduled Auxiliary Control (SAC-X). Using this new learning technique, complex behaviors can be learned from scratch with sparse rewards. Via off-policy RL methods, this agent attempted to learn a set of general auxiliary tasks simultaneously. Better execution and scheduling allowed the agent to explore the environment efficiently in the presence of sparse rewards.
Autonomous vehicles	(Na et al., 2022)	A DDPG algorithm with a novel sampling method called Highlight Experience Replay with rapid training is presented for collision avoidance in autonomous vehicles.
	(Fan et al., 2022)	For unmanned surface vehicles, a DQN-based RL collision avoidance (RLCA) algorithm is presented.
	(Lin et al., 2021)	Presented a decentralized repositioning framework based on DRL for effective obstacle avoidance and collision avoidance in unmanned aerial vehicle networks.
Robotics	(Zhu et al., 2022)	A rule-based reinforcement learning (RuRL) algorithm is proposed for autonomous efficient robot navigation with reduced sample complexity and time cost.
Large scale optimization	(Wang et al., 2022)	For large-scale optimization, the RL level-based particle swarm optimization algorithm (RLLPSO) is proposed.

(*continued on next page*)

Table 5 (*continued*)

Application area	Reference	Short description
Energy management	(Ding et al., 2022a)	A multi-agent DRL algorithm based on target value competition (MADRL-TVC) is presented to solve the problem of economic load dispatch under varied conditions.
Network security	(Caminero et al., 2019)	A novel algorithm for intrusion detection was proposed, with an incredibly fast artificial neural network-based classifier providing high prediction performance.

Table 6
Modern DNN architecture integration with RL framework.

Application area	DNN architecture	Reference	Short description
Natural Language Processing (NLP)	Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019)	(Xiao et al., 2022)	A novel framework, FusionSum, is proposed that mimics human behavior in summarizing multiple sentences.
		(Wang et al., 2019)	For text summarization, a unified extractive-abstractive approach is proposed.
		(Vo, 2022)	HINRL4Rec, a novel personal recommender system, is presented.
	Long Short TermMemory (LSTM) (Hochreiter & Schmidhuber, 1997) and BERT	(Gharagozlou, et al., 2022)	An attention mechanism-based approach RLAS-BIABC is presented for answer selection.
	Graph Neural Network (GNN) (Scarselli et al., 2009)	(Chen et al., 2019b)	AgentGraph, a GNN-based task-oriented voiced dialogue system, is introduced for global and local decision-making.
Energy management	Convolutional neural network (CNN) (Krizhevsky et al., 2012)	(Claessens et al., 2018)	A novel CNN-based Residential Load Control approach with automated state-time feature extraction is introduced.
	Generative Adversarial Network (GAN) (Goodfellow et al., 2014)	(Fang et al., 2021)	To resolve the problem of renewable energy accommodation in a hierarchical microgrid, a temporal difference and GAN-based DRL framework is proposed.
	LSTM	(Liu et al., 2020)	Proposed a novel bidirectional LSTM network-based parallel RL approach for developing an energy management system for a hybrid tracked vehicle.
		(Lu et al., 2022)	A novel LSTM-based demand response strategy for smart facility energy management was introduced in order to reduce electricity expenditures while maintaining a satisfaction index.
		(Zhang et al., 2021)	Introduced an LSTM-based approach for learning the optimal charging strategy for electric vehicles in order to reduce the user's charging expense.
	Auto encoder	(Zhang et al., 2019)	For energy-efficient management in real-time embedded devices, a stacked auto encoder-based DQN model is presented.
Classification	Boltzmann machine (Salakhutdinov & Hinton 2009)	(Harney et al., 2020)	For multipartite qubit systems, a framework based on quantum entanglement and machine learning is proposed.
	CNN & LSTM	(Khayyat & Elrefaei, 2022)	Established an LSTM-based RL agent capable of interacting with historical low-quality manuscript pictures and retrieving the pictures that are most relevant to a query picture.
Manufacturing system/ Process control	GNN	(Huang et al., 2021)	A collaborative control strategy based on GNN is proposed for optimizing individual machining process parameters in response to process and system-level conditions.
		(Huang et al., 2022)	A control framework for machine and process-level manufacturing production optimization is proposed.
Resource allocation (communication network)	GNN	(Li & Zhu, 2022)	HRLOrch, a paradigm for enabling online virtual network function service chains provisioning in elastic optical data center interconnections, is introduced.
		(Sun et al., 2021)	DeepOpt, a data-driven technique for solving the virtual network function placement problem in distinct network topologies, is introduced.
		(Sun et al., 2020a)	DeepMigration is presented as a solution to the flow migration problem and to improve service quality in network function virtualization elastic control.
	GAN	(Hua et al., 2020)	A GAN-DDQN model is presented to find the best solution for demand-aware resource management in network slicing by combining deep distributional RL and GAN.
Robotics	LSTM	(Samsani & Muhammad, 2021)	A novel DRL framework for socially compliant secure and safe social robot navigation is proposed.
		(Apuroop et al., 2021)	Based on the DRL framework, developed a complete area coverage management mechanism for a honeycomb-shaped robot.

(*continued on next page*)

Table 6 (*continued*)

Application area	DNN architecture	Reference	Short description
	GAN	(Wu et al., 2020a)	Proposed a robotic calligraphy model for directly learning to draw Chinese character strokes using pictures taken from Chinese calligraphic textbooks.
Intelligent vehicles	GAN	(Naeem et al., 2021)	A GAN-based DRL technique is proposed for learning the optimum intelligent transmission scheduling policies in the Cognitive Internet of Vehicles. When compared to the classical DQN, GAN is used for approximating the action value distribution and preventing the negative impact of noise and randomness on the reward.
	CNN, LSTM & GAN	(Chen et al., 2019a)	A unique DRL model that can learn from both simulated and actual traffic environments is presented and utilized to make end-to-end planning decisions in emergency situations with reduced decision-making errors.
	Recurrent neural network (RNN) (Schuster & Paliwal 1997) & LSTM	(Cao et al., 2022)	Presented a learning based vehicle tracking system for autonomous vehicles experiencing Li-DAR failure under varying lighting conditions.
Healthcare	U-Net (Ronneberger, Fischer, & Brox, 2015)	(Bi et al., 2022)	Presented a simulation-based solution for automatically navigating an ultrasound probe to a vascular standardized surface for improved image stability and reproducibility.
		(Liu et al., 2022)	Proposed the 3D spatio temporal U-Net DRL framework for medical video analysis.
	LSTM	(Ståhl et al., 2019)	A novel approach is presented for learning how to modify and improve molecules in order for them to have desirable features for boosting efficiency and quality in drug discovery.
	LSTM & RNN	(Banerjee & Singh, 2021)	Using a RL approach, an LSTM-RNN-based prediction methodology for missing data segments in an ECG signal is presented.
Routing problem	Graph convolutional encoder & Attention decoder network (Vaswani et al., 2017)	(Luo et al., 2022b)	To solve the travelling salesman problem, a novel DRL-based encoder-decoder architecture called GCE-MAD Net is developed.
	RNN	(Nazari et al., 2018)	A DRL-based end-to-end solution for handling the vehicle routing problem is proposed.
	LSTM	(Kyaw et al., 2020)	Presented a novel methodology to optimize coverage path planning on the traveling salesman problem using the DRL framework.
	Encoder-decoder & LSTM	(Zhao et al., 2021)	Presented a novel DRL framework for resolving the vehicle routing problem.

off-policy DRL algorithms, which is then utilized for training the RL agent before more data is gathered. However, these off-policy methods are usually unsuccessful unless the provided dataset is correlated with the present policy. As a result, it is generally desirable to have data collected by a highly controlled method, such as a human operator or a well-monitored environment. In general, the decision between on-policy, off-policy, and offline RL methods depends on the problem being solved and available resources such as time, data, and processing capacity.

6.1. Some recent research work in RL

Table 5 includes information on significant research work and novel RL algorithms introduced in various disciplines in recent years.

6.2. Modern DNN architecture integration with RL framework

The major application areas of RL have been board games, computer games, robotic tasks in simulation, such as manipulation and locomotion (Mnih et al., 2015; Lillicrap et al., 2016; Silver et al., 2016; Silver et al., 2017a; Silver et al., 2018) along with call admission control (Marbach et al., 1998), traffic light control (Houli et al., 2010; Abdoos et al., 2011), gas turbine control (Schaefer et al., 2007), customer interaction (Silver et al., 2013), news recommendation systems (Li et al., 2010), energy sector (Anderson et al., 2011), stocks trading (Moody & Saffell, 2001), clinical decision-making (Shortreed et al., 2011).

RL is also quite effective for resolving stochastic sequential decision-making problems. Today, RL is widely used to solve challenging problems in interdisciplinary domains where problems may be modeled as sequential decision-making problems with states, actions, and possible rewards. There are numerous challenges with automating human strategy design, and RL is one possible approach. In recent years, the number of RL applications in many new fields has increased drastically (Nguyen et al., 2017; Arulkumaran et al., 2017; Buşoniu et al., 2018; Li, 2018; Liu et al., 1907; Luong et al., 2019; Li et al., 2019, Khan et al., 2021) including interdisciplinary domains, such as combinatorial optimization (Wang et al., 2023; Mazyavkina et al., 2021), recommender systems (Liu et al., 2023; Afsar et al., 2022), graph data mining (Mingshuo et al., 2022), network security (Adawadkar & Kulkarni, 2022; Caminero et al., 2019), manufacturing (Li et al., 2023), process control (Nian et al., 2020; Shin et al., 2019), resource allocation (Li & Zhu, 2022), NLP (Xiao et al., 2022), music (Jaques et al., 2017), drawing (Ha & Eck, 2017), mathematics (Pan et al., 2018; Zeng et al., 2022), chemistry (Rajak et al., 2021; Segler et al., 2018), biology (Jumper et al., 2021; Mahmud et al., 2018) to resolve difficult stochastic sequential decision-making problems.

In earlier work, mostly convolutional and feed-forward networks were used in DRL applications (Lillicrap et al., 2016; Mnih et al., 2015; Silver et al., 2016; Silver et al., 2017a; Silver et al., 2018), but in recent years, advanced or modern deep neural network (DNN) architectures are enabling the use of RL in previously under-explored areas such as Natural Language Processing (NLP), energy management, resource allocation, process control, healthcare and routing problems (Zhou et al., 2021; Zhao et al., 2021; Xiao et al., 2022; Radoglou-Grammatikis et al., 2022).

Although various advanced DNN architectures with numerous designs, depths, and sizes have shown amazing achievement in the field of NLP and computer vision (He et al., 2016; Radford et al., 2019) that deal

Fig. 7. Some promising research fields in RL.

with complex image inputs, the use of modern DNN architecture in the field of RL remains relatively under-explored. Less attention has been paid to the field of exploring modern DNN architecture designs in DRL for policy and value networks or state feature extraction.

The exploration of modern DNN architectures in RL can be advantageous in achieving two of the essential goals of RL agents: sampling efficiency and generalization of RL agents for various similar tasks. Based on the excellent results obtained with modern artificial neural network architectural choices in areas similar to computer vision, deeper networks and denser connections for RL have been under investigation in recent years. Table 6 shows some recent RL works in which the possibilities of using modern DNN in policy and value networks or state feature extraction were investigated for various application fields.

6.3. *Promising research fields in RL*

RL has made some significant advances in the field of AI but still has a lot of issues. To make RL more adaptable and sophisticated for solving more general problems, different research fields need to be investigated more deeply. Many promising research fields exist in RL that could lead to a more evolved and scalable RL paradigm. Fig. 7 shows some promising research fields in RL and Table 7 gives some details of these highly rich and appealing research fields. While some of the research areas covered in Table 7 have previously been extensively studied, some still require further in-depth research.

6.4. *Multi agent reinforcement learning (MARL)*

Multi agent reinforcement learning (MARL) is one of the most important research fields in RL because it can address many real-world complex collaborative decision-making problems. In the earlier discussion in this review, only single agent RL algorithms are discussed. A single RL agent interacting with the environment cannot fully solve a large number of control problems; however, multiple RL agents interacting with the same environment at the same time can achieve more efficient learning. MARL is focused on the analysis and control of the behaviors of multiple RL agents that exist in a shared environment. Each agent learns from its own reward function and acts on its own interests; these individual interests may be conflicting with the interests of some of the other agents, resulting in complicated group interaction. MARL adds one more complexity to the RL problem, which is the introduction of nonstationarity due to the change in the behaviors of other agents in the environment after learning (Buşoniu et al., 2008). An efficient communication network between the agents can be an effective approach for improved cooperation and learning in MARL. A successful attempt for better communication in MARL between the agents has been made (Foerster et al., 2016a; Peng et al., 2017; Sukhbaatar et al., 2016). MARL has been a fascinating study topic that has received a lot of attention for applications in computer gaming and robotics (Peng et al., 2017; Berner et al., 2019; Scheikl et al., 2021). Researchers have proposed fantastic MARL algorithms in recent years, which have found their applicability in a variety of application fields. Through a multi-agent hide-and-seek problem and standard RL learning algorithms, the OpenAI research team found that agents can generate a self-supervised evolving action strategy (Baker et al., 2019). In the environment, they found six evolving phases in the agent's counteraction strategy against opponents. The agents were quite successful in exploiting the environment's shortcomings to achieve better rewards. Fig. 8 summarizes various significant MARL algorithms based on actor-critic, policy optimization, and value-based methodologies (Dai, Yu, Wang, & Baldi, 2022; L. Ding, Lin, Shi, & Yan, 2022; R. Ding, Xu, Gao, & Shen, 2022; Kar, Moura, & Poor, 2013; Matta et al., 2019; Omidshafiei, Pazis, Amato, How, & Vian, 2017; Zhang, Yang, Liu, Zhang, & Basar, 2018; Bloembergen et al., 2010; Foerster et al., 2016b; Foerster et al., 2018; Foerster et al., 2017; Gupta et al., 2017; Iqbal and Sha, 2019; Lowe et al., 2020; Matignon et al., 2007; Palmer et al., 2018; Rashid et al., 2020; Rashid et al., 2018; Sukhbaatar et al., 2016; Sunehag et al., 2018).

6.5. *Action shaping*

Along with the promising research fields in RL outlined in Table 7, one of the fundamental research fields that can enhance RL agent learning in complex problems with a large action space is action shaping. In RL framework, standard action spaces such as continuous, discrete, and multi-discrete are generally used. Multi-discrete action space is an extension of discrete action space in which each action is a vector of independent discrete actions. In some cases, the action space

Table 7
Overview of some promising research fields in RL.

Research fields	Some details and related work
Model-based RL	• With the help of pixel information, the system dynamics model can be learned using several DRL methods (Oh et al., 2015; Wahlström et al., 2015; Watter et al., 2015). • If the agent can learn a satisfactorily precise environment model, then the agent can be controlled directly by simple control strategies from camera images. • By improving the data efficiency, the application of DNN in model-based RL can be encouraged. • A thorough review of model-based RL research to date is provided in detail in Moerland et al. (2022) and Luo et al. (2022a), covering both conventional methods and more recent advancements.
Imitation learning	• This approach in the traditional RL literature is known as behavioral cloning, where we demonstrate the optimal action strategy by an expert to the agent. This approach can broaden DRL's applicability to a wider range of real-world tasks in which the agent must learn in a real-world setting. • Using human-generated driving data some exciting results were obtained with the autonomous car ALVINN (Pomerleau, 1989). • Google DeepMind research team produced some amazing results with Deep Q-learning from Demonstrations (DQfD), where they used a small set of demonstration data to accelerate the learning process (Hester et al., 2018).
Inverse RL	• Inverse RL estimates the unknown reward function of an agent from a given optimal policy or expert demonstration (Ng & Russell, 2000). • These nonlinear reward functions can be estimated using DNN. A combination of RL and IRL can be used effectively to improve learning (Argall et al., 2009).
Hierarchical RL	• Sparse rewards can lead the RL agent to poor exploration and learning. Hierarchical RL (HRL) (Barto & Mahadevan, 2003; Sutton & Singh, 1999; Dietterich, 2000) is one of the effective solutions for this problem, where we divide the problem into hierarchical subtasks. • HRL relies on hierarchies of sub-policies and decision-making is not required at each time step. • In HRL, a two-level policy structure (Kulkarni et al., 2016) is used. Lower-level or sub-policies are trained with traditional RL algorithms to suggest the action, while the higher-level policies are trained to select these sub-policies over a complete task. • In Pateria et al. (2021), a comprehensive overview of the HRL research to date is offered in detail, ranging from the traditional methods to the most recent developments.
Transfer learning	• Transfer learning is one of the exciting concepts in DL, which suggests that a DNN architecture developed for a specific learning problem can be reused as the initial point of reference for another task by simply changing the ending layers of DNN. The starting layers of DNN can be easily generalized for a new task. • This exciting feature of DNN can be utilized effectively in DRL to solve the data inefficiency problem (Parisotto et al., 2016). Transfer learning can also accelerate the learning process in a new task if the task has some similarity with the previously learned tasks (Barreto et al., 2017).
Sim to real transfer	• The physical reality gap between the real-world and the simulator holds back the RL agents from achieving stable performance in real-world operations. • MIT research team developed their photorealistic data-driven simulator (VISTA) (Amini et al., 2020) for DRL-based autonomous vehicles. Their trained agent was quite successful in test drives for previously unseen roads. • A Soft Actor-Critic based navigation framework was proposed in Chaffre et al. (2020), which drastically reduced the additional training required before real-world application.
Meta RL	• Developing agents that can quickly adapt to new tasks by utilizing knowledge gained through prior experience with similar tasks is a significant challenge in RL. • Learning to learn rather than acting in the environment is the main focus of *meta*-RL (Wang et al., 2016a). • Meta-RL, which learns an exploration methodology that gathers data for adaptation also learns a policy that swiftly adjusts to a new similar task (Kirsch et al., 2020; Pong et al., 2022).
Offline RL	• Offline RL is a ML technique that allows an agent to learn from a fixed dataset of previously acquired experience rather than interacting with the environment in real-time. • Direct learning from real-world encounters, for example, can be time-consuming, costly, and possibly dangerous in robotics or autonomous driving applications. In such circumstances, offline RL can assist in training the agent by exploiting existing datasets. • Offline RL has outperformed many state-of-the-art approaches in model-based (Kidambi et al., 2020; Yu et al., 2020; Swazinna et al., 2021) and model-free (Kalyanakrishnan & Stone 2007; Fujimoto et al., 2019) scenarios.
Constrained RL	• The fundamental objective of RL agents is to learn a policy in a manner that allows them to maximize cumulative reward. But sometimes, the learned behavior might be detrimental to the agent or adjacent people. Constrained RL (Achiam et al., 2017; Chow et al., 2017) is an important research area in which the agent has some constrained settings along with the cost functions that prevent the agents from taking dangerous actions. • Constrained RL may prove to be more beneficial than conventional RL for ensuring that agents adhere to safety standards (HasanzadeZonuzy et al., 2021).
Safe policy improvement	• In Batch RL, Safe Policy Improvement (Laroche et al., 2019; Nadjahi et al., 2019; Scholl et al., 2023) is a crucial area of research. • With the help of this approach, the RL agent can learn a policy from a fixed dataset, and that policy will always perform at least as well as the one that was used to gather the data.
Interpretable RL	• Interpretable RL (Glanois et al., 2022; Alharin et al., 2020; Verma et al., 2018; Hein et al., 2017; Maes et al., 2012), which aims to produce interpretable and verifiable agent policies, is another fascinating area of RL research. • For safety concerns, a learned agent policy must be interpretable before being used in critical fields, including robotics, autonomous vehicles, medical applications, etc. Interpretability ensures that field professionals can review and validate a proposed policy before it is implemented.
Efficiency improvement	• The vast amount of interaction data that current reinforcement learning approaches still need could make them too expensive to use in practical applications. • In order to use RL techniques to solve real-world practical problems, improving sample efficiency (Wang et al., 2017; Yarats et al., 2021) is a crucial area for advancement. • Different research directions to reduce the sample cost of reinforcement learning are discussed from various aspects in Yu (2018).

can be a combination of continuous, discrete, or multi-discrete action spaces.

To minimize attempting "unnecessary" actions and to simplify the implementation of the RL approach in computer games, the action space for these environments is often reduced in comparison to the game's original action space. Such action space modifications, known as action space shaping, are common in the RL approach, particularly in video games where the action space is too large for learning environments (Berner et al., 2019; Vinyals et al., 2019). Some common action-shaping strategies for the RL agent include the discretization of continuous actions, combining several actions into one action, and eliminating some actions (Gao et al., 2018; Vinyals et al., 2019; Berner et al., 2019; Guss et al., 2021).

Currently, this complex action space transformation is done instinctively, with minimal thorough research to support design decisions. Investigations examining specific transformations to the action spaces have not been included in the associated research works. Such investigations may decrease the number of initial learning trials. When an agent is unable to learn a task, it is typically challenging to pinpoint whether the problem is with the training algorithm, the action space, the

reward function, the state space, or the environment dynamics. Therefore, one of these options can be eliminated by action space transformation research, which will facilitate learning.

6.6. RL evaluation platforms

Due to the high sample cost and safety issues, it is not practical to train or test the RL agent directly in the real- world. For a problem such as game playing, we can easily evaluate our algorithm in the game environment. However, we need some evaluation platforms to evaluate and compare the performance of RL algorithms for more difficult sequential decision-making tasks to solve real-life problems. Fig. 9 provides an overview of the important evaluation platforms available for various control problems (Bellemare, Naddaf, Veness, & Bowling, 2013; Brockman et al., 2016; Beattie et al., 2016; Castro et al., 2018; Fu et al., 2021; Tassa et al., 2018).

Amazon SageMarker, Microsoft Project Malmo (Johnson et al., 2016), PyGame Learning Environment (PLE), AI Safety Gridworlds (Leike et al., 2017), StarCraft II Learning Environment (SC2LE) (Vinyals et al., 2017), and RL-Glue (Tanner & White, 2009) are some of the other RL evaluation platforms.

The majority of the platforms discussed here are only useful for a few specific problems. We need richer platforms to bridge the gap between simulation and real-world performance. New realistic evaluation platforms or simulators for complicated real-world problems must be developed in order to improve learning and performance.

7. Conclusions

This paper only covers ground-breaking work and important algorithms in order to give a broad overview of RL and discusses future research prospects in this fascinating field. A list of various important RL materials, such as books, thesis, conferences, and journal papers, are included in the bibliography of this review for ready reference of the readers.

RL has achieved some incredible outcomes with the help of DL. Before comprehending DRL, it is necessary to have a thorough grasp of RL. DL's success in both academia and industry has risen dramatically in recent years. However, one common criticism of DL is that it operates in a black box mode, making it difficult to understand how it works. This should not be used as an argument against DL; rather, a greater understanding of how DL works will be beneficial to both DL and RL communities in general.

Based on transfer, imitation, supervised and semi-supervised

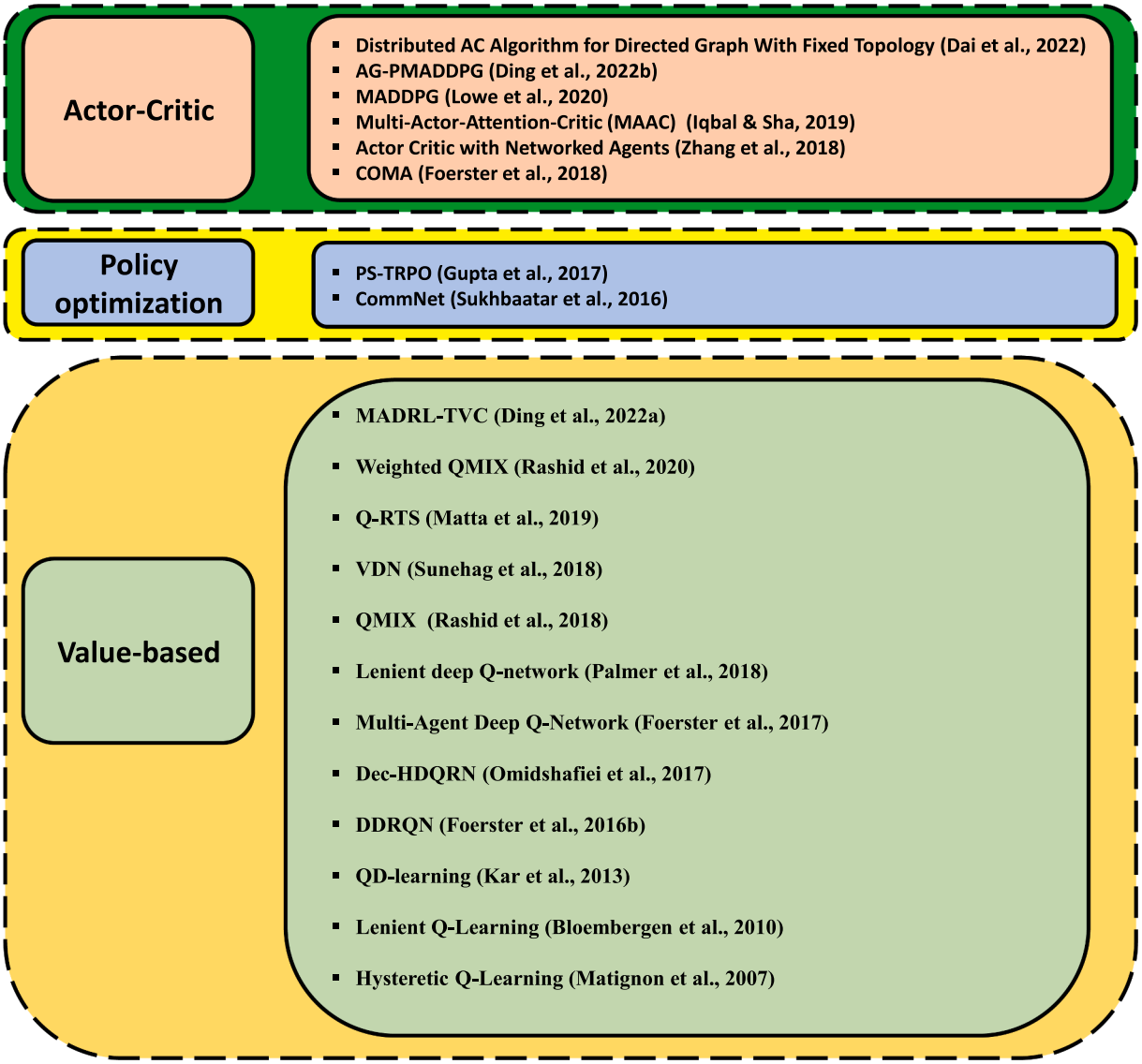

Fig. 8. MARL algorithms summary.

Fig. 9. Overview of some RL evaluation platforms.

learning, various new RL mechanisms have emerged to enhance the speed and quality of learning, and more such mechanisms will emerge in the future. The RL methodologies still have a lot of room for improvement in order to make them scalable to real-world challenges. RL has a lot of potential in real-world problems, but it has had very little success so far. To manage huge amounts of input data associated with real-world challenges, better DRL algorithms are required. Another major challenge for advancement in this field is the computational power required for DRL. As a result, only large companies and institutes are doing major work in this field.

Stability, convergence, scalability, accuracy, learning speed, sample efficiency, simplicity, robustness, and safety are all important considerations for RL algorithms. In order to enhance this field, it is necessary to evaluate criticisms and remarks about the performance of various RL algorithms so that proper research endeavor is made to overcome the same.

CRediT authorship contribution statement

Ashish Kumar Shakya: Writing – original draft, Conceptualization, Writing – review & editing, Visualization. **Gopinatha Pillai:** Supervision, Conceptualization, Writing – review & editing. **Sohom Chakrabarty:** Supervision, Conceptualization, Writing – review & editing, Visualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

Abdoos, M., Mozayani, N., & Bazzan, A. L. C. (2011). Traffic light control in non-stationary environments based on multi agent Q-learning. In *Proceedings of the 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (pp. 1580-1585).

Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)* (pp. 22–31).

Adawadkar, A. M. K., & Kulkarni, N. (2022). Cyber-security and reinforcement learning — A brief survey. *Engineering Applications of Artificial Intelligence, 114*, 1–18.

Afsar, M. M., Crump, T., & Far, B. (2022). Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys, 55*(7), 1–38.

Agrawal, S. & Jia, R. (2017). Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)* (pp. 1184-1194).

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., & Zhang, L. (2019). Solving Rubik's cube with a robot hand. arXiv preprint arXiv: 1910.07113.

Alharin, A., Doan, T.-N., & Sartipi, M. (2020). Reinforcement learning interpretation methods: A survey. *IEEE Access, 8*, 171058–171077.

Amini, A., Gilitschenski, I., Phillips, J., Moseyko, J., Banerjee, R., Karaman, S., & Rus, D. (2020). Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters, 5*(2), 1143–1150.

Amini, A., & Soleimany, A. (2020). Introduction to deep learning. *MIT Course Number, 6*, S191.

Anderson, R. N., Boulanger, A., Powell, W. B., & Scott, W. (2011). Adaptive stochastic control for the smart grid. In *Proceedings of the IEEE, 99(6)* (pp. 1098-1115).

Apuroop, K. G. S., Le, A. V., Elara, M. R., & Sheu, B. J. (2021). Reinforcement learning-based complete area coverage path planning for a modified htrihex robot. *Sensors, 21*(4), 1–20.

Aradi, S. (2022). Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions On Intelligent Transportation Systems, 23*(2), 740–759.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems, 57*(5), 469–483.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 26–38.

Arwa, E. O., & Folly, K. A. (2020). Reinforcement learning techniques for optimal power control in grid-connected microgrids: A comprehensive review. *IEEE Access, 8*, 208992–209007.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally Weighted Learning. *Artificial Intelligence Review, 11*, 11–73.

Azar, M. G., Osband, I., & Munos, R. (2017). Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (PMLR)* (pp. 263-272).

Azar, N. A., Shahmansoorian, A., & Davoudi, M. (2020). From inverse optimal control to inverse reinforcement learning: A historical review. *Annual Reviews in Control, 50*, 119–138.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., & Blundell, C. (2020a). Agent57: Outperforming the atari human benchmark. arXiv preprint arXiv: 2003.13350v1.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., & Blundell, C. (2020b). Never give up : Learning directed exploration strategies. arXiv preprint arXiv:2002.06038.

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., & Mordatch, I. (2019). Emergent tool use from multi-agent autocurricula. arXiv preprint arXiv: 1909.07528.

Bakhtin, A., Wu, D. J., Lerer, A., Gray, J., Jacob, A. P., Farina, G., Miller, A. H., & Brown, N. (2022). Mastering the game of no-press diplomacy via human-regularized reinforcement learning and planning. arXiv preprint arXiv:2210.05492v1.

Banerjee, S., & Singh, G. K. (2021). Deep neural network based missing data prediction of electrocardiogram signal using multiagent reinforcement learning. *Biomedical Signal Processing and Control, 67*(102508), 1–9.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Van Hasselt, H., & Silver, D. (2017). Successor features for transfer in reinforcement learning. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)* (pp. 501–510).

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications, 13*, 41–77.

Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research, 15*, 319–350.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., & Petersen, S. (2016). DeepMind Lab. arXiv preprint arXiv: 1612.03801.

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML 70)* (pp. 449-458).

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

Bellemare, M. G., Veness, J., & Bowling, M. (2012). Investigating Contingency Awareness Using Atari 2600 Games. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence* (pp. 864–871).

Bellman, R. E. (1956). A Problem in the sequential design of experiments. *The Indian Journal of Statistics, 16*(3/4), 221–229.

Bellman, R. E. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics, 6*(5), 679–684.

Bellman, R. E. (1958). Dynamic programming and stochastic control processes. *Information and Control, 1*(3), 228–239.

Bellman, R. E. (1972). *Dynamic programming.* Princeton University Press.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P. P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H, P. d. O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., & Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.

Bertsekas, D. P. (2005). *Dynamic programming and optimal control* (vol. 1). Belmont, MA: Athena Scientific.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming.* Belmont, MA: Athena Scientific.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., & Lee, M. (2009). Natural actor-critic algorithms. *Automatica, 45*(11), 2471–2482.

Bi, Y., Jiang, Z., Gao, Y., Wendler, T., Karlas, A., & Navab, N. (2022). VesNet-RL: Simulation-based reinforcement learning for real-world US probe navigation. *IEEE Robotics and Automation Letters, 7*(3), 6638–6645.

Bloembergen, D., Kaisers, M., & Tuyls, K. (2010). Lenient frequency adjusted Q-learning. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence (BNAIC 2010)* (pp. 19–26).

Bradtke, S. J., & Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning, 22*, 33–57.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym.* arXiv preprint arXiv:1606.01540.

Buşoniu, L., Babuška, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 38*(2), 156–172.

Buşoniu, L., de Bruin, T., Tolić, D., Kober, J., & Palunko, I. (2018). Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control, 46*, 8–28.

Caminero, G., Lopez-martin, M., & Carro, B. (2019). Adversarial environment reinforcement learning algorithm for intrusion detection. *Computer Networks, 159*, 96–109.

Campbell, M., Hoane, A. J., & Hsu, F. H. (2002). Deep blue. *Artificial Intelligence, 134*, 57–83.

Cao, M., Wang, R., Chen, N., & Wang, J. (2022). A learning-based vehicle trajectory-tracking approach for autonomous vehicles with lidar failure under various lighting conditions. *IEEE/ASME Transactions on Mechatronics, 27*(2), 1011–1022.

Castro, P. S., Moitra, S., Gelada, C., Kumar, S., & Bellemare, M. G. (2018). Dopamine: A research framework for deep reinforcement learning. arXiv preprint arXiv: 1812.06110.

Chaffre, T., Moras, J., Chan-Hon-Tong, A., & Marzat, J. (2020). Sim-to-real transfer with incremental environment complexity for reinforcement learning of depth-based robot navigation. In *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics* (pp. 314–323).

Chen, L., Chen, Z., Tan, B., Long, S., Gasic, M., & Yu, K. (2019). AgentGraph: Toward universal dialogue management with structured deep reinforcement learning. *IEEE/ACM Transactions on Audio Speech and Language Processing, 27*(9), 1378–1391.

Chen, L., Hu, X., Tian, W., Wang, H., Cao, D., & Wang, F. Y. (2019). Parallel planning: A new motion planning framework for autonomous driving. *IEEE/CAA Journal of Automatica Sinica, 6*(1), 236–246.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., & Mordatch, I. (2021). Decision transformer : Reinforcement learning via sequence modeling. arXiv preprint arXiv: 2106.01345.

Chen, X., Qu, G., Tang, Y., Low, S., & Li, N.a. (2022). Reinforcement learning for selective key applications in power systems: Recent advances and future challenges. *IEEE Transactions On Smart Grid, 13*(4), 2935–2958.

Chow, Y., Ghavamzadeh, M., Janson, L., & Pavone, M. (2017). Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research, 18*(1), 6070–6120.

Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of the 33nd Conference on Advances in Neural Information Processing Systems* (pp. 4754-4765).

Ciosek, K., Vuong, Q., Loftin, R., & Hofmann, K. (2019). Better exploration with optimistic actor-critic. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)* (pp. 1787–1798).

Claessens, B. J., Vrancx, P., & Ruelens, F. (2018). Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control. *IEEE Transactions on Smart Grid, 9*(4), 3259–3269.

Crites, R. H., & Barto, A. G. (1994). An actor / critic algorithm that equivalent to Q-learning. In *Proceedings of the 7th International Conference on Neural Information Processing Systems* (pp. 401–408).

Dai, P., Yu, W., Wang, H., & Baldi, S. (2022). Distributed actor-critic algorithms for multiagent reinforcement learning over directed graphs. *IEEE Transactions On Neural Networks and Learning Systems*, 1–12.

Dayan, P. (1992). The convergence of TD(λ) for general λ. *Machine Learning, 8*, 341–362.

Morais, D.e., Gustavo, A. P., Marcos, L. B., Bueno, J. N. A. D., de Resende, N. F., Terra, M. H., & Grassi, V., Jr. (2020). Vision-based robust control framework based on deep reinforcement learning applied to autonomous ground vehicles. *Control Engineering Practice, 104*, 104630.

Deisenroth, M. P., & Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (pp. 465-472).

Devlin, J., Chang, M., Kenton, L., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1* (pp. 4171–4186).

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research, 13*, 227–303.

Ding, L., Lin, Z., Shi, X., & Yan, G. (2022). Target-value-competition-based multi-agent deep reinforcement learning algorithm for distributed nonconvex economic dispatch. *IEEE Transactions on power systems*, 1–14.

Ding, R., Xu, Y., Gao, F., & Shen, X. (2022). Trajectory design and access control for air – Ground coordinated communications system with multiagent deep reinforcement learning. *IEEE Internet of Things Journal, 9*(8), 5785–5798.

Du, W., & Ding, S. (2020). A survey on multi-agent deep reinforcement learning: From the perspective of challenges and applications. *Artificial Intelligence Review, 54*(12), 3215–3238.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 2001–2014).

Duguleana, M., & Mogan, G. (2016). Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Systems With Applications, 62*, 104–115.

Elmo: Computer Shogi Association, Results of the 27th world computer shogi championship. (2023). Retrieved from http://www2.computer-shogi.org/wcsc27/index_e.html. Accessed March 10, 2023.

Engel, Y., Mannor, S., & Ron, M. (2005). Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning* (pp. 201–208).

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research, 6*, 503–556.
Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., & Kavukcuoglu, K. (2018). IMPALA: Scalable distributed Deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning ICML* 80 (pp. 1407–1416).
Fakoor, R., Chaudhari, P., Soatto, S., & Smola, A. J. (2020). META-Q-Learning. arXiv preprint arXiv:1910.00125.
Fan, Y., Sun, Z., & Wang, G. (2022). A novel reinforcement learning collision avoidance algorithm for usvs based on maneuvering characteristics and COLREGs. *Sensors, 22*(6), 1–29.
Fang, D., Guan, X., & Peng, Y. (2021). Distributed deep reinforcement learning for renewable energy accommodation assessment with communication uncertainty in internet of energy. *IEEE Internet Of Things Journal, 8*(10), 8557–8569.
Farahmand, A. M., Ghavamzadeh, M., Szepesvári, C., & Mannor, S. (2008). Regularized policy iteration. In *Proceedings of the Advances in Neural Information Processing Systems 21* (pp. 441–448).
Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., … Kohli, P. (2022). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature, 610*, 47–53.
Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., & Levine, S. (2018). Model-based value expansion for efficient model-free reinforcement learning. arXiv preprint arXiv: 1803.00101v1.
Foerster, J. N., Assael, Y. M., De Freitas, N., & Whiteson, S. (2016a). Learning to communicate with deep multi-agent reinforcement learning. In *Proceedings of the 29th International Conference on Advances in Neural Information Processing Systems* (pp. 2145–2153).
Foerster, J. N., Assael, Y. M., Freitas, N. de, & Whiteson, S. (2016b). Learning to communicate to solve riddles with deep distributed recurrent Q-networks. arXiv preprint arXiv:1602.02672.
Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (pp. 2–7).
Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H. S., Kohli, P., & Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)* (pp. 1146-1155).
Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I, Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., & Legg, S. (2017). Noisy networks for exploration. arXiv preprint arXiv: 1706.10295v3.
Fox, R., Pakman, A., & Tishby, N. (2016). Taming the noise in reinforcement learning via soft updates. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence* (pp. 202–211).
Fu, K. S. (1970). Learning control systems—Review and outlook. *IEEE Transactions on Automatic Control, 15*, 210–221.
Fu, Q., Han, Z., Chen, J., Lu, Y., Wu, H., & Wang, Y. (2022). Applications of reinforcement learning for building energy efficiency control: A review. *Journal of Building Engineering, 50*, 1–22.
Fu, J., Kumar, A., Nachum, O., Tucker, G., & Levine, S. (2021). D4RL: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv: 2004.07219v4.
Fujimoto, S., Meger, D., & Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine (PMLR 97) (pp. 2052-2062)*.
Fujimoto, S., Van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 2587–2601).
Gao, Y., Xu, H., Lin, Ji., Yu, F., Levine, S., & Darrell, T. (2018). Reinforcement learning from imperfect demonstrations. arXiv preprint arXiv: 1802.05313.
García, J., & Shafie, D. (2020). Teaching a humanoid robot to walk faster through safe reinforcement learning. *Engineering Applications of Artificial Intelligence, 88*(1), 103360.
Gharagozlou, H., Mohammadzadeh, J., Bastanfard, A., & Ghidary, S. S. (2022). RLAS-BIABC: A reinforcement learning-based answer selection using the bert model boosted by an improved ABC algorithm. *Computational Intelligence and Neuroscience*, 1–21.
Glanois, C., Weng, P., Zimmer, M., Li, D., Yang, T., Hao, J., & Liu, W. (2022). A survey on interpretable reinforcement learning. arXiv preprint arXiv: 2112.13112v2.
Goodfellow, I. J., Pouget-abadie, J., Mirza, M., Xu, B., Warde-farley, D., Ozair, S., Courville. A., & Bengio, Y. (2014). Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems 2 (NIPS'14)* (pp. 2672–2680).
Gronauer, S., & Diepold, K. (2022). Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review, 55*, 895–943.
Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems* (pp. 66–83).
Guss, W. H., Castro, M. Y., Devlin, S., Houghton, B., Kuno, N. S., Loomis, C., Milani, S., Mohanty, S., Nakata, K., Salakhutdinov, R., Schulman, J., Shiroshita, S., Topin, N., Ummadisingu, A., & Vinyals, O. (2021). NeurIPS 2020 Competition : The MineRL competition on sample efficient reinforcement learning using human priors. arXiv preprint arXiv:2101.11071.
Ha, D., & Eck, D. (2017). A neural representation of sketch drawings. arXiv preprint arXiv:1704.03477v4.
Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 1352–1361).
Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy Maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning 5* (pp. 1861–1870).
Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., & Levine, S. (2018b). Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.
Hafner, D., Lillicrap, T., Ba, J., & Norouzi, M. (2020). Dream to control: Learning behaviors by latent imagination. arXiv preprint arXiv: 1912.01603v3.
Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering diverse domains through world models. arXiv preprint arXiv:2301.04104v1.
Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Pfaff , T., Weber, T., Buesing, L., & Battaglia, P. W. (2020). Combining Q-learning and search with amortized value estimates. *International Conference on Learning Representations (ICLR 2020)*..
Hansen, N., & Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariancematrix adaptation. In *Proceedings of the IEEE International Conference on Evolutionary Computation* (pp. 312–317).
Harney, C., Pirandola, S., Ferraro, A., & Paternostro, M. (2020). Entanglement classification via neural network quantum states. *New Journal of Physics, 22*(4), 1–13.
HasanzadeZonuzy, A., Bura, A., Kalathil, D., & Shakkottai, S. (2021). Learning with safety constraints: Sample complexity of reinforcement learning for constrained MDPs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence* (pp. 7667–7674).
Hasselt, H. V. (2010). Double Q-learning. In *Proceedings of the Advances in Neural Information Processing Systems, 23* (pp. 2613–2621).
Hasselt, H. Van, Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-Learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence, (AAAI-16)* (pp. 2094–2100).
Hausknecht, M., Lehman, J., Miikkulainen, R., & Stone, P. (2014). A neuroevolution approach to general atari game playing. *IEEE Transactions on Computational Intelligence and AI in Games, 6*(4), 355–366.
Haykin, S. (2008). *Neural Networks and Learning Machines*. Pearson: Prentice Hall.
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision* (pp. 630–645).
Hein, D., Hentschel, A., Runkler, T., & Udluft, S. (2017). Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies. *Engineering Applications of Artificial Intelligence, 65*, 87–98.
Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (pp. 3207–3214).
Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *32nd AAAI Conference on Artificial Intelligence (AAAI)* (pp. 3215–3222).
Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., … Gruslys, A. (2018). Deep Q-learning from demonstrations. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (pp. 3223–3230).
Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735–1780.
Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Hasselt, H., & Silver, D. (2018). Distributed prioritized experience replay. In *Proceedings of the International Conference on Learning Representations (ICLR 2018)*..
Houli, D., Zhiheng, L., & Yi, Z. (2010). Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network. *EURASIP Journal on Advances in Signal Processing*, 1–7.
Howard, R. (1960). *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press.
Hu, Y., Hua, Y., Liu, W., & Zhu, J. (2021). Reward shaping based federated reinforcement learning. *IEEE Access, 9*, 67259–67267.
Hua, Y., Li, R., Zhao, Z., Zhang, H., & Chen, X. (2020). GAN-based deep distributional reinforcement learning for resource management in network slicing. In *Proceedings of the 2019 IEEE Global Communications Conference* (pp. 1–6).
Huang, J., Su, J., & Chang, Q. (2022). Graph neural network and multi-agent reinforcement learning for machine-process-system integrated control to optimize production yield. *Journal of Manufacturing Systems, 64*, 81–93.
Huang, J., Zhang, J., Chang, Q., & Gao, R. X. (2021). Integrated process-system modelling and control through graph neural network and reinforcement learning. *CIRP Annals, 70*(1), 377–380.
Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning (ICML)* (pp. 448–456).
Iqbal, S., & Sha, F. (2019). Actor-attention-critic for multi-agent reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, PMLR 97* (pp. 2961–2970).
Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., & Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. arXiv preprint arXiv:1611.05397.
Jaksch, T., Ortner, R., & Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research, 11*(4), 1563–1600.
Jaques, N., Gu, S., Bahdanau, D., Herńandez-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, 70 (pp. 1645-1654)*..
Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P. A., Strouse, D. J., … Freitas, N. D. (2019). Social influence as intrinsic motivation for multi-agent deep

reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 3040–3049).
Jiang, R., Zahavy, T., Xu, Z., White, A., Hessel, M., Blundell, C., & Hasselt, H. Van. (2021). Emphatic algorithms for deep reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning (PMLR 139)* (pp. 5023–5033).
Johnson, M., Hofmann, K., Hutton, T., & Bignell, D. (2016). The malmo platform for artificial intelligence experimentation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (pp. 4246–4247).
Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature, 596*(7873), 583–589.
Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan. D., Finn. C., Kozakowski. P., Levine. S., Mohuuddin. A., Sepassi. R., Tucker. G., & Michalewski, H. (2020). Model based reinforcement learning for atari. arXiv preprint arXiv:1903.00374.
Kalweit, G., & Boedecker, J. (2017). Uncertainty driven imagination for continuous deep reinforcement learning. In *Proceedings of the 1st Annual Conference on Robot Learning,78* (pp. 195-206).
Kalyanakrishnan, S., & Stone, P. (2007). Batch reinforcement learning in a complex domain. In *Proceedings of the 6th International Joint Conference On Autonomous Agents And Multiagent Systems* (pp. 650–657).
Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., & Dabney, W. (2019). Recurrent experience replay in distributed reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR 2019)*..
Kar, S., Moura, J. M. F., & Poor, H. V. (2013). QD-Learning : A collaborative distributed strategy for multi-agent reinforcement learning through. *IEEE Transactions on Signal Process, 61*(7), 1848–1862.
Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Li, F. F. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 1725–1732).
Khan, M. F., Gazara, R. K., Nofal, M. M., Chakrabarty, S., Dannoun, E. M. A., AL-Hmouz, R., & Mursaleen, M. (2021). Reinforcing synthetic data for meticulous survival prediction of patients suffering from left ventricular systolic dysfunction. *IEEE Access, 9*, 72661–72669.
Khan, M. A., Khan, M. R. J., Tooshil, A., Sikder, N., Mahmud, M. A. P., Kouzania, Z., & Nahid, A. (2020). A systematic review on reinforcement learning-based robotics within the last decade. *IEEE Access, 8*, 176598–176623.
Khayyat, M. M., & Elrefaei, L. A. (2022). Deep reinforcement learning approach for manuscripts image classification and retrieval. *Multimedia Tools and Applications, 81*(11), 15395–15417.
Kidambi, R., Rajeswaran, A., Netrapalli, P., & Joachims, T. (2020). MOReL: Model-based offline reinforcement learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*..
Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., & Pérez, P. (2022). Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions On Intelligent Transportation Systems, 23*(6), 4909–4926.
Kirsch, L., Steenkiste, S. Van, & Schmidhuber, J. (2020). Improving generalization in meta reinforcement learning using learned objectives. arXiv preprint arXiv: 1910.04098.
Klopf, A. H. (1972). Brain function and adaptive systems: A heterostatic theory, Technical Report, Air Force Cambridge Research Labs Hanscom AFB MA.
Klopf, A. H. (1975). A comparison of natural and artificial intelligence. *ACM SIGART Bulletin*, 11–13.
Klopf, A. H. (1982). *The hedonistic neuron: A theory of memory, learning, and intelligence.* Washington: Hemisphere Pub. Corp.
Kobayashi, T., & Sugino, T. (2020). Reinforcement learning for quadrupedal locomotion with design of continual–hierarchical curriculum. *Engineering Applications of Artificial Intelligence, 95*, 103869.
Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research, 32*(11), 1238–1274.
Kohl, N., & Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation* (ICRA 2004) (pp. 2619–2624).
Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in Neural Information Processing Systems 12* (pp. 1008–1014).
Konda, V. R., & Tsitsiklis, J. N. (2003). On actor-critic algorithms. *SIAM Journal on Control and Optimization, 42*(4), 1143–1166.
Krishnan, S., Garg, A., Liaw, R., Thananjeyan, B., Miller, L., Pokorny, F. T., & Goldberg, K. (2019). SWIRL : A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *The International Journal of Robotics Research, 38*(2–3), 126–145.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems 1* (pp.1097–1105).
Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., & Tenenbaum, J. B. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 29th International Conference on Neural Information Processing Systems* (pp. 3682–3690).
Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative Q-learning for offline reinforcement learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)* (pp. 1–13).
Kyaw, P. T., Paing, A., Thu, T. T., Mohan, R. E., Le, A. V., & Veerajagadheswar, P. (2020). Coverage path planning for decomposition reconfigurable grid-maps using deep reinforcement learning based travelling salesman problem. *IEEE Access, 8*, 225945–225956.
Ladosz, P., Ben-iwhiwhu, E., Dick, J., Ketz, N., Kolouri, S., Krichmar, J. L., … Soltoggio, A. (2022). Deep reinforcement learning with modulated Hebbian plus Q-network architecture. *IEEE Transactions on Neural Networks and Learning Systems, 33*(5), 2045–2056.
Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research, 4*(6), 1107–1149.
Laroche, R., Trichelair, P., & Combes, R. T. D. (2019). Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning (PMLR 97)* (pp. 3652-3661).
Lazaric, A., Ghavamzadeh, M., & Munos, R. (2012). Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research, 13*, 3041–3074.
Le, N., Rathour, V. S., Yamazaki, K., Luu, K., & Savvides, M. (2022). Deep reinforcement learning in computer vision: A comprehensive survey. *Artificial Intelligence Review, 55*(4), 2733–2819.
Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.
Lee, A. X., Nagabandi, A., Abbeel, P., & Levine, S. (2020). Stochastic latent actor-critic : Deep reinforcement learning with a latent variable model. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)* (pp. 1–12).
Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Lefrancq, L., & Legg, S. (2017). AI safety gridworlds. arXiv preprint arXiv:1711.09883.
Levine, S., & Abbeel, P. (2014). Learning neural network policies with guided policy search under unknown dynamics. In *Proceedings of the 27th International Conference on Neural Information Processing Systems (vol. 1)* (pp. 1071-1079).
Levine, S., & Koltun, V. (2013). Guided policy search. In *Proceedings of the 30th International Conference on Machine Learning, PMLR 28(3)* (pp. 1-9).
Li, Y. (2018). Deep reinforcement learning. arXiv preprint arXiv:1810.06339v1.
Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., … Vinyals, O. (2022). Competition-level code generation with AlphaCode. *Science, 378*(6624), 1092–1097.
Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web (WWW '10)* (pp. 661-670).
Li, G., Gomez, R., Nakamura, K., & He, B. (2019). Human-centered reinforcement learning: A survey. *IEEE Transactions on Human-Machine Systems, 49*(4), 337–349.
Li, W., & Todorov, E. (2004). Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics* (pp. 222–229).
Li, C., Zheng, P., Yin, Y., Wang, B., & Wang, L. (2023). Deep reinforcement learning in smart manufacturing: A review and prospects. *CIRP Journal of Manufacturing Science and Technology, 40*, 75–101.
Li, B., & Zhu, Z. (2022). GNN-based hierarchical deep reinforcement learning for NFV-oriented online resource orchestration in elastic optical DCIs. *Journal of Lightwave Technology, 40*(4), 935–946.
Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., & Wierstra, D., (2016). Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations ICLR* (pp. 1–14).
Lin, J., Chiu, H., & Gau, R. (2021). Decentralized planning-assisted deep reinforcement learning for collision and obstacle avoidance in UAV networks. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*.
Lin, L. J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning, 8*, 293–321.
Liu, F., & Qian, C. (2021). Prediction guided meta-learning for multi-objective reinforcement learning. In *2021 IEEE Congress on Evolutionary Computation (CEC)*.
Liu, H., Cai, K., Li, P., Qian, C., Zhao, P., & Wu, X. (2023). REDRL: A review-enhanced deep reinforcement learning model for interactive recommendation. *Expert Systems With Applications, 213*, 1–13.
Liu, J., & Feng, L. (2021). Diversity evolutionary policy deep reinforcement learning. *Computational Intelligence and Neuroscience*, 2021.
Liu, T., Meng, Q., Huang, J. J., Vlontzos, A., Rueckert, D., & Kainz, B. (2022). Video summarization through reinforcement with a 3D spatio-temporal U-net. *IEEE Transactions on Image Processing, 31*, 1573–1586.
Liu, S., Ngiam, K. Y., & Feng, M. (2019). Deep reinforcement learning for clinical decision support: A brief survey. arXiv preprint arXiv: 1907.09475.
Liu, T., Tian, B., Ai, Y., & Wang, F. Y. (2020). Parallel reinforcement learning-based energy efficiency improvement for a cyber-physical system. *IEEE/CAA Journal of Automatica Sinica, 7*(2), 617–626.
Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2020). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing* (pp. 6383–6393).
Lu, R., Bai, R., Luo, Z., Jiang, J., Sun, M., & Zhang, H. T. (2022). Deep reinforcement learning-based demand response for smart facilities energy management. *IEEE Transactions on Industrial Electronics, 69*(8), 8554–8565.
Luo, J., Li, C., Fan, Q., & Liu, Y. (2022b). A graph convolutional encoder and multi-head attention decoder network for TSP via reinforcement learning. *Engineering Applications of Artificial Intelligence, 112* (2022)104848, 1–16.
Luo, F., Xu, T., Lai, H., Chen, X., Zhang, W., & Yu, Y. (2022a). A survey on model-based reinforcement learning. arXiv preprint arXiv:2206.09328v1.
Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y. C., & Kim, D. I. (2019). Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys and Tutorials, 21*(4), 3133–3174.
Maei, H. R., Szepesvari, C., Bhatnagar, S., Precup, D., Silver, D., & Sutton, R. S. (2009). Convergent temporal-difference learning with arbitrary smooth function approximation. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems* (pp. 1204–1212).
Maes, F., Fonteneau, R., Wehenkel, L., & Ernst, D. (2012). Policy search in a space of simple closed-form formulas: towards interpretability of reinforcement learning. In:

Ganascia, JG., Lenca, P., Petit, JM. (eds) *Discovery Science. DS 2012*. Lecture Notes in Computer Science, vol 7569. Springer, Berlin, Heidelberg.
Mahmud, M., Kaiser, M. S., Hussain, A., & Vassanelli, S. (2018). Applications of deep learning and reinforcement learning to biological data. *IEEE Transactions on Neural Networks and Learning Systems, 29*(6), 2063–2079.
Marbach, P., Mihatsch, O., & Tsitsiklis, J. N. (1998). Call admission control and routing in integrated services networks using reinforcement learning. In *Proceedings of the 37th IEEE Conference on Decision and Control, (vol.1)* (pp. 563-568).
MathWorks, Block diagram of reinforcement learning. (2023). Retrieved from https://www.mathworks.com/help/reinforcement-learning/ug/create-simulink-environments-for-reinforcement-learning.html. Accessed March 10, 2023.
Matignon, L., Laurent, G. J., & Fort-piat, N. Le. (2007). Hysteretic Q-Learning : an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 64–69).
Matta, M., Cardarilli, G. C., Nunzio, L. D., Fazzolari, R., Giardino, D., Re, M., … Spanò, S. (2019). Q-RTS : A real-time swarm intelligence based on multi-agent Q-learning. *Electronics Letters, 55*(10), 589–591.
Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research, 134*(1), 1–15.
Melo, F. S., Meyn, S. P., & Ribeiro, M. I. (2008). An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th International Conference on Machine Learning, 5* (pp. 664–671).
Mendel, J. M. (1966). A survey of learning control systems. *ISA Transactions, 5*, 297–303.
Mendonca, R., Gupta, A., Kralev, R., Abbeel, P., Levine, S., & Finn, C. (2019). Guided meta-policy search. In *Proceedings of the 33rd Conference on Neural Information Processing Systems 32* (pp. 1–12).
Michie, D., & Chambers, R. A. (1968). BOXES, An experiment in adaptive control. *Machine Intelligence, 2*, 137–152.
Miljković, Z., Mitić, M., Lazarević, M., & Babić, B. (2013). Neural network reinforcement learning for visual control of robot manipulator. *Expert Systems With Applications, 40*, 1721–1736.
Mingshuo, N., Dongming, C., & Dongqi, W. (2022). Reinforcement learning on graph: A survey. arXiv preprint arXiv:2204.06127v3.
Minsky, M. (1961). Steps Toward Artificial Intelligence. *Proceedings of the IRE, 49*(1), 8–30.
Mnih, V., Badia, A. P., Mirza, L., Graves, A., Harley, T., Lillicrap, T. P., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML) 48* (pp. 1928–1937).
Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv: 1312.5602.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518* (7540), 529–533.
Moerland, T. M., Broekens, J., Plaat, A., & Jonker., C. M. (2022). Model-based reinforcement learning: A Survey. arXiv preprint arXiv: 2006.16712v4.
Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Network, 12*(4), 875–889.
Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning, 13*(1), 103–130.
Munos, R., Stepleton, T., Harutyunyan, A., & Bellemare, M. G. (2016). Safe and efficient off-policy reinforcement learning. In *Proceedings of the 30th Conference on Neural Advances in Neural Information Processing Systems* (pp. 1054–1062).
Na, S., Niu, H., Lennox, B., & Arvin, F. (2022). Bio-inspired collision avoidance in swarm systems via deep reinforcement learning. *IEEE Transactions on Vehicular Technology, 71*(3), 2511–2526.
Nadjahi, K., Laroche, R., & Combes, R. T. (2019). Safe policy improvement with soft baseline bootstrapping. arXiv preprint arXiv: 1907.05079v1.
Naeem, F., Seifollahi, S., Zhou, Z., & Tariq, M. (2021). A generative adversarial network enabled deep distributional reinforcement learning for transmission scheduling in internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems, 22*(7), 4550–4559.
Nagabandi, A., Kahn, G., Fearing, R. S., & Levine, S. (2017). Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. arXiv preprint arXiv: 1708.02596v2.
Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., Maria, A. D., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., Legg, S., Mnih, V., Kavukcuoglu, K., & Silver, D. (2015). Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296v2.
Narasimhan, K., Kulkarni, T. D., & Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1–11).
Nazari, M., Oroojlooy, A., Snyder, L. V., & Takáč, M. (2018). Reinforcement learning for solving the vehicle routing problem. arXiv preprint arXiv:1802.04240.
Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., & Liang, E. (2006). Autonomous inverted helicopter flight via reinforcement learning. Experimental Robotics IX. *Springer Tracts in Advanced Robotics, 21*, 363–372.
Ng, A., & Russell, S. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference on Machine Learning* (pp. 663–670).
Nguyen, N. D., Nguyen, T., & Nahavandi, S. (2017). System design perspective for human-level agents using deep reinforcement learning: A survey. *IEEE Access, 5*, 27091–27102.
Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions On Cybernetics, 50*(9), 3826–3839.
Nian, R., Liu, J., & Hunag, B. (2020). A review On reinforcement learning: Introduction and applications in industrial process control. *Computers and Chemical Engineering, 139*, 1–30.
Noaeen, M., Naik, A., Goodman, L., Crebo, J., Abrar, T., Abad, Z. S. H., … Far, B. (2022). Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Systems With Applications, 199*, 1–32.
Oh, J., Guo, X., Lee, H., Lewis, R., & Singh, S. (2015). Action-conditional video prediction using deep networks in Atari games. In *28th International Conference on Neural Information Processing Systems* (pp. 2863–2871).
Oh, J., Singh, S., & Lee, H. (2017). Value prediction network. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)* (pp. 6118-6128).
Omidshafiei, S., Pazis, J., Amato, C., How, J. P., & Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 2681–2690).
Ormoneit, D., & Sen, S. (2002). Kernel-based reinforcement learning. *Machine Learning, 49*, 161–178.
Paine, T. L., Paduraru, C., Michi, A., Gulcehre, C., Żołna, K., Novikov, A., Wang. Z., & Freitas, N. de. (2020). Hyperparameter selection for offline reinforcement learning. arXiv preprint arXiv:2007.09055.
Palmer, G., Tuyls, K., Bloembergen, D., & Savani, R. (2018). Lenient multi-agent deep reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)* (pp. 443–451).
Pan, Y., Farahmand, A., White, M., Nabi, S., Grover, P., & Nikovski, D. (2018). Reinforcement learning with function-valued action spaces for partial differential equation control. In *Proceedings of the 35th International Conference on Machine Learning* (pp. 3986–3995).
Pane, Y. P., Nageshrao, S. P., Kober, J., & Babuška, R. (2019). Reinforcement learning based compensation methods for robot manipulators. *Engineering Applications of Artificial Intelligence, 78*, 236–247.
Parisotto, E., Ba, J., & Salakhutdinov, R. (2016). Actor-mimic deep multitask and transfer reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)* (pp. 1–16).
Pateria, S., Subagdja, B., Tan, A., & Quek, C. (2021). Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Survey, 54*(5), 1–35.
Pavlov, I. P., & Anrep, G. V. (1927). *Conditioned reflexes: An investigation of the physiological activity of the cerebral cortex*. London: Oxford University Press.
Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., & Wang, J. (2017). Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play StarCraft combat games. arXiv preprint arXiv: 1703.10069.
Peters, J., Mulling, K., & Altun, Y. (2010). Relative entropy policy search. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence, 24(1)* (pp. 1607-1612)..
Peters, J., & Schaal, S. (2007). Applying the episodic natural actor-critic architecture to motor primitive learning. In *Proceedings of the 15th European Symposium on Artificial Neural Networks (ESANN 2007)* (pp. 295-300).
Peters, J., & Schaal, S. (2008a). Natural actor-critic. *Neurocomputing, 71*, 1180–1190.
Peters, J., & Schaal, S. (2008b). Reinforcement learning of motor skills with policy gradients. *Neural Networks, 21*(4), 682–697.
Polydoros, A. S., & Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on Robotics. *Journal of Intelligent and Robotic Systems: Theory and Applications, 86*(2), 153–173.
Pomerleau, D. A. (1989). ALVINN: An autonomous land vehicle in a neural network. In *Proceedings of the 1st International Conference on Advances in Neural Information Processing Systems* (pp. 305–313).
Pong, V., Gu, S., Dalal, M., & Levine, S. (2018). Temporal difference models: Model-free deep RL for model-based control. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)* (pp. 1–14).
Pong, V. H., Nair, A., Smith, L., Huang, C., & Levine, S. (2022). Offline meta-reinforcement learning with online self-supervision. arXiv preprint arXiv: 2107.03974v4.
Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. OpenAI: Technical report.
Radoglou-Grammatikis, P., Rompolos, K., Sarigiannidis, P., Argyriou, V., Lagkas, T., Sarigiannidis, A., … Wan, S. (2022). Modeling, detecting, and mitigating threats against industrial healthcare systems: A combined software defined networking and reinforcement learning approach. *IEEE Transactions on Industrial Informatics, 18*(3), 2041–2052.
Rajak, P., Krishnamoorthy, A., Mishra, A., Kalia, R., Nakano, A., & Vashishta, P. (2021). Autonomous reinforcement learning agent for chemical vapor deposition synthesis of quantum materials. *npj Computational Materials, 7*(108).
Rakelly, K., Zhou, A., Quillen, D., Finn, C., & Levine, S. (2019). *Efficient off-policy meta-reinforcement learning via probabilistic context variables, 97*, 5331–5340.
Rasheed, F., Yau, K. A., Noor, R. M., Wu, C., & Low, Y. (2020). Deep reinforcement learning for traffic signal control: A review. *IEEE Access, 8*, 208016–208044.
Rashid, T., Farquhar, G., Peng, B., & Whiteson, S. (2020). Weighted QMIX : Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)* (pp. 10199–10210).
Rashid, T., Samvelyan, M., Witt, C. S. de, Farquhar, G., Foerster, J., & Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (PMLR 80)* (pp. 4295–4304).

Rawlik, K., Toussaint, M., & Vijayakumar, S. (2012). On stochastic optimal control and reinforcement learning by approximate inference. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence 8* (pp. 353–360).

Riedmiller, M. (1999). Concepts and facilities of a neural reinforcement learning control architecture for technical process control. *Neural Computing and Applications, 8*(4), 323–338.

Riedmiller, M. (2005). Neural fitted Q iteration - First experiences with a data efficient neural reinforcement learning method. In *Proceedings of the 16th European Conference on Machine Learning (ECML) 3720 LNAI* (pp. 317–328).

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van De Wiele, T., & Springenberg, T. (2018). Learning by playing - Solving sparse reward tasks from scratch. In *Proceedings of the 35th International Conference on Machine Learning (ICML) 10* (pp. 4344–4353).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. arXiv preprint arXiv arXiv:1505.04597.

Rudin, N., Hoeller, D., Reist, P., & Hutter, M. (2021). Learning to walk in minutes using massively parallel deep reinforcement learning. arXiv preprint arXiv:2109.11978.

Rummery, G. A., & Niranjan, M. (1994). On-Line Q-Learning using connectionist systems. In *Cambridge University Engineering Department, Cambridge England* (pp. 1–20).

Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann Machines. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, PMLR 5* (pp. 448–455).

Samsani, S. S., & Muhammad, M. S. (2021). Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning. *IEEE Robotics and Automation Letters, 6*(3), 5223–5230.

Samuel, A. L. (1959). Some studies in machine learning using the game of Chekers. *IBM Journal of Research and Development, 3*(3), 210–229.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks, 20*(1), 61–80.

Schaefer, A. M., Schneegass, D., Sterzing, V., & Udluft, S. (2007). A neural reinforcement learning approach to gas turbine control. In *Proceedings of the 2007 International Joint Conference on Neural Networks* (pp. 1691-1696).

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)* (pp. 1–21).

Scheikl, P. M., Gyenes, B., Davitashvili, T., Younis, R., Schulze, A., Muller-Stich, B. P., Neumann. G., & Mathis-Ullrich, F. (2021). Cooperative assistance in robotic surgery through multi-agent reinforcement learning. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1859–1864).

Schmitt, S., Hessel, M., & Simonyan, K. (2019). Off-policy actor-critic with shared experience replay. arXiv preprint arXiv:1909.11583.

Scholl, P., Dietrich, F., Otte, C., & Udluft, S. (2023). Safe policy improvement approaches and their limitations. In *Proceedings of the Agents and Artificial Intelligence: 14th International Conference, ICAART 2022* (pp. 74–98).

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., & Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature, 588*(7839), 604–609.

Schulman, J. (2016). *Optimizing Expectations: From deep reinforcement learning to stochastic computation graphs*. Berkeley: Dept. of Computer Science University of California. Ph. D. thesis.

Schulman, J., Levine, S., Moritz, P., Jordan, M., & Abbeel, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning, (PMLR) 37* (pp. 1889–1897).

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)* (pp. 1–14).

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

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing, 45*(11), 2673–2681.

Segler, M. H. S., Preuss, M., & Waller, M. P. (2018). Planning chemical syntheses with deep neural networks and symbolic AI. *Nature, 555*, 604–610.

Shannon, C. E. (1950). XXII. Programming a computer for playing chess. *Philosophical Magazine and Journal of Science, 41*(314), 256–275.

Shannon, C. E. (1952). "Theseus" maze-solving mouse. Retrieved from http://cyberneticzoo.com/mazesolvers/1952-—theseus-maze-solving-mouse-—claude-shannon-american/. Accessed March 10, 2023.

Shin, J., Badgwell, T. A., Liu, K., & Lee, J. H. (2019). Reinforcement learning –Overview of recent progress and implications for process control. *Computers and Chemical Engineering, 127*, 282–294.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., & Murphy, S. A. (2011). Informing sequential clinical decision-making through reinforcement learning: An empirical study. *Machine Learning, 84*(1–2), 109–136.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., van den Driessche, G., … Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., … Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*, 1140–1144.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning, 32* (pp. 387-395).

Silver, D., Newnham, L., Barker, D., Weller, S., & McFall, J. (2013). Concurrent reinforcement learning from customer interactions. In *Proceedings of the 30th International Conference on Machine Learning, PMLR,* 28(3) (pp. 924-932).

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature, 550*(7676), 354–359.

Silver, D., van Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A., & Degris, T. (2017b). The predictron: End-to-end learning and planning. In *Proceedings of the 34th International Conference on Machine Learning* (*vol. 70)* (pp. 3191-3199).

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)* (pp. 1–14).

Singh, S. P., Jaakkola, T., & Jordan, M. I. (1994). Reinforcement learning with soft state aggregation. *Advances in Neural Information Processing Systems*, *7* (pp. 361–368).

Singh, B., Kumar, R., & Singh, V. P. (2022). Reinforcement learning in robotic applications: A comprehensive survey. *Artificial Intelligence Review, 55*, 945–990.

Singh, S., Man, D. L., Kearns, M., & Walker, M. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research, 16*, 105–133.

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning, 22*, 123–158.

Soleymani, F., & Paquet, E. (2021). Deep graph convolutional reinforcement learning for financial portfolio management – DeepPocket. *Expert Systems With Applications, 182*, 115127.

Song, L., Li, D., Wang, X., & Xu, X. (2022). AdaBoost maximum entropy deep inverse reinforcement learning with truncated gradient. *Information Sciences, 602*, 328–350.

Srinivas, A., Jabri, A., Abbeel, P., Levine, S., & Finn, C. (2018). Universal planning networks. arXiv preprint arXiv:1804.00645.

Ståhl, N., Falkman, G., Karlsson, A., Mathiason, G., & Boström, J. (2019). Deep reinforcement learning for multiparameter optimization in de novo drug design. *Journal of Chemical Information and Modeling, 59*(7), 3166–3176.

Stockfish: Strong open source chess engine. (2022). Retrieved from https://stockfishchess.org/. Accessed March 10, 2023.

Strehl, A. L., Lihong, L., Wiewiora, E., Langford, J., & Littman, M. L. (2006). PAC model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)* (pp. 881–888).

Subramanian, A., Chitlangia, S., & Baths, V. (2022). Reinforcement learning and its connections with neuroscience and psychology. *Neural Networks, 145*, 271–287.

Sukhbaatar, S., Szlam, A., & Fergus, R. (2016). Learning multiagent communication with backpropagation. In *Proceedings of the 29th International Conference on Neural Information Processing Systems (NIPS)* (pp. 2252–2260).

Sun, P., Lan, J., Li, J., Guo, Z., & Hu, Y. (2021). Combining deep reinforcement learning with graph neural networks for optimal VNF placement. *IEEE Communications Letters, 25*(1), 176–180.

Sun, P., Lan, J., Li, J., Guo, Z., Hu, Y., & Hu, T. (2020). Efficient flow migration for NFV with Graph-aware deep reinforcement learning. *Computer Networks, 183*, 107575.

Sun, P., Zhou, W., & Li, H. (2020b). Attentive experience replay. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, 34(04)* (pp. 5900–5907).

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot. M., Sonnerat. N., Leibo. J. Z., Tuyls. K., & Graepel, T. (2018). Value-decomposition networks for cooperative multi-agent learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS'18* (pp. 2085–2087).

Sutton, R. S. (1978). Single channel theory: A neuronal theory of learning. *Brain Theory Newsletter, 3*, 72–75.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference Machine Learning Proceedings* (pp. 216–224).

Sutton, R. S., & Barto, A. G. (1981a). An adaptive network that constructs and uses an internal model of its world. *Cognition and Brain Theory, 4*(3), 217–246.

Sutton, R. S., & Barto, A. G. (1981b). Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review, 88*(2), 135–170.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to Reinforcement Learning* (1st ed). Cambridge, MA, USA: MIT Press.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning An Introduction*. Cambridge, MA, USA: MIT Press.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems, 12*, 1057–1063.

Sutton, R. S., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence, 112*(1999), 181–211.

Swazinna, P., Udluft, S., & Runkler, T. (2021). Overcoming model bias for robust offline deep reinforcement learning. arXiv preprint arXiv:2008.05533v4.

Szita, I., & Lorincz, A. (2006). Learning tetris using the noisy cross-entropy method. *Neural Computation, 18*(12), 2936–2941.

Tanner, B., & White, A. (2009). RL-Glue: Language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research, 10*, 2133–2136.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. de Las, Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., & Riedmiller, M. (2018). DeepMind Control Suite. arXiv preprint arXiv:1801.00690.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communication of the ACM, 38*(3), 58–67.

Thanh, H. V., An, L. T. H., & Chien, B. D. (2008). Maximum entropy inverse reinforcement learning brian. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence* (pp. 1433–1438).

Thorndike, E. L. (1911). *Animal intelligence*. New York: Macmillan Company.

Tsitsiklis, J. N., & Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control, 42*, 674–690.
Turing, A. (1948). Intelligent machinery: Report for National physical laboratory universal turing machine.
Van Seijen, H., & Sutton, R. S. (2014). True online TD(λ). In *Proceedings of the 31st International Conference on Machine Learning (ICML) 32* (pp. 1048–1056).
Van Seijen, H., van Hasselt, H., Whiteson, S., & Wiering, M. (2009). A theoretical and empirical analysis of expected sarsa. In *Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (IEEE)* (pp. 177–184).
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 5998–6008).
Verma, A., Murali, V., Singh, R., Kohli, P., & Chandhuri, S. (2018). Programmatically interpretable reinforcement learning. *Proceedings of the 35th International Conference on Machine Learning (PMLR)*.
Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., … Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature, 575*, 350–354.
Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., Hasselt, H. V., Silver, D., Lillicrap, T., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J., & Tsing, R. (2017). StarCraft II: A new challenge for reinforcement learning. arXiv preprint arXiv: 1708.04782.
Vo, T. (2022). An integrated network embedding with reinforcement learning for explainable recommendation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications, 26*(8), 3757–3775.
Wahlström, N., Schön, T. B., & Deisenroth, M. P. (2015). From pixels to torques: Policy learning with deep dynamical models. arXiv preprint arXiv: 1502.02251.
Waltz, M. D., & Fu, K. S. (1965). A heuristic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control, 10*, 390–398.
Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & De Freitas, N. (2017). Sample efficient actor-critic with experience replay. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)* (pp. 1–20).
Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., & Botvinick, M. (2016a). Learning to reinforcement learn. arXiv preprint arXiv: 1611.05763v3.
Wang, Q., Lai, K. H., & Tang, C. (2023). Solving combinatorial optimization problems over graphs with BERT-Based deep reinforcement learning. *Information Sciences, 619*, 930–946.
Wang, Q., Liu, P., Zhu, Z., Yin, H., Zhang, Q., & Zhang, L. (2019). A text abstraction summary model based on BERT word embedding and reinforcement learning. *Applied Sciences, 9*(21), 1–19.
Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)* (pp. 1995–2003).
Wang, F., Wang, X., & Sun, S. (2022). A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization. *Information Sciences, 602*, 298–312.
Watkins, C. J. C. H. (1989). Learning from delayed rewards, King's College Cambridge, Ph.D. thesis.
Watter, M., Springenberg, J. T., Boedecker, J., & Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)* (pp. 2746–2754).
Wayne, G., Hung, C. C., Amos, D., Mirza, M., Ahuja, A., Barwinska, A. G., Rae, J., Mirowski, P., Leibo, J. Z., Santoro, A., Gemici, M., Reynolds, M., Harley, T., Abramson, J., Mohamed, S., Rezende, D., Saxton, D., Cain, A., Hillier, C., Silver, D., Kavukcuoglu, K., Botvinick, M., Hassabis, D., & Lillicrap, T. (2018). Unsupervised predictive memory in a goal-directed agent. arXiv preprint arXiv: 1803.10760.
Weber, T., Racanière, S., Reichert, D. P., Buesing, L., Guez,A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., & Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning. arXiv preprint arXiv: 1707.06203v2.
Werbos, P. J. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems, XXI I, 1977*, 25–38.
Willia, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*(3), 229–256.
Witten, I. H. (1977). An adaptive optimal controller for discrete-time markov environments. *Information and Control, 34*, 286–295.
Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences, 538*, 142–158.
Wu, Y., Mansimov, E., Liao, S., Grosse, R., & Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Proceddings of the 31st Conference on Neural Information Processing Systems* (pp. 5285–5294).
Wu, R., Zhou, C., Chao, F., Yang, L., Lin, C. M., & Shang, C. (2020). Integration of an actor-critic model and generative adversarial networks for a Chinese calligraphy robot. *Neurocomputing, 388*, 12–23.
Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., & Sumner, A. (2013). *TORCS, The open racing car simulator, v1.3.5*, 2013.
Xiao, L., He, H., & Jin, Y. (2022). FusionSum: Abstractive summarization with sentence fusion and cooperative reinforcement learning. *Knowledge-Based Systems, 243* (108483), 1–10.
Xu, X., Zuo, L., & Huang, Z. (2014). Reinforcement learning algorithms with function approximation: Recent advances and applications. *Information Sciences, 261*, 1–31.
Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., & Fergus, R. (2021). Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI-21)* (pp. 10674–10681).
Yin, L., Chen, L., Liu, D., Huang, X., & Gao, F. (2021). Quantum deep reinforcement learning for rotor side converter control of double-fed induction generator-based wind turbines. *Engineering Applications of Artificial Intelligence, 106*, 104451.
Yu, Y. (2018). Towards sample efficient reinforcement learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-18)* (pp. 5739–5743).
Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J., Levine, S., Finn, C., & Ma, T. (2020). MOPO: Model-based offline policy optimization. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*.
Yu, L., Zhang, C., Jiang, J., Yang, H., & Shang, H. (2021). Reinforcement learning approach for resource allocation in humanitarian logistics. *Expert Systems With Applications, 173*(2), 114663.
Zanette, A. & Brunskill, E. (2019). Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *Proceedings of the 36th International Conference on Machine Learning (PMLR 97)* (pp. 7304-7312).
Zeng, S., Cai, Y., & Zou, Q. (2022). Deep neural networks based temporal-difference methods for high-dimensional parabolic partial differential equations. *Journal of Computational Physics, 468*(2022).
Zhang, L.i., Gao, Y., Zhu, H., & Tao, L. (2022). A distributed real-time pricing strategy based on reinforcement learning approach for smart grid. *Expert Systems With Applications, 191*, 116285.
Zhang, Q., Lin, M., Yang, L. T., Chen, Z., & Li, P. (2019). Energy-efficient scheduling for real-time systems based on deep Q-learning model. *IEEE Transactions on Sustainable Computing, 4*(1), 132–141.
Zhang, F., Yang, Q., & An, D. (2021). CDDPG: A deep-reinforcement-learning-based approach for electric vehicle charging control. *IEEE Internet of Things Journal, 8*(5), 3075–3087.
Zhang, K., Yang, Z., Liu, H., Zhang, T., & Basar, T. (2018). Fully decentralized multi-agent reinforcement learning with networked agents. In *Proceedings of the 35th International Conference on Machine Learning, PMLR 80* (pp. 5872–5881).
Zhao, J., Mao, M., Zhao, X., & Zou, J. (2021). A hybrid of deep reinforcement learning and local search for the vehicle routing problems. *IEEE Transactions on Intelligent Transportation Systems, 22*(11), 7208–7218.
Zhao, J., Zhou, W., Zhao, T., Zhou, Y., & Li, H. (2020). State representation learning for effective deep reinforcement learning. In *IEEE International Conference on Multimedia and Expo. (ICME)* (pp. 1–6).
Zhou, Y., Van Kampen, E. J., & Chu, Q. (2019). Hybrid hierarchical reinforcement learning for online guidance and navigation with partial observability. *Neurocomputing, 331*, 443–457.
Zhou, S. K., Le, H. N., Luu, K., Nguyen, V. H., & Ayache, N. (2021). Deep reinforcement learning in medical imaging: A literature review. *Medical Image Analysis, 73* (102193), 1–20.
Zhu, Y., Wang, Z., Chen, C., & Dong, D. (2022). Rule-based reinforcement learning for efficient robot navigation with space reduction. *IEEE/ASME Transactions on Mechatronics, 27*(2), 846–857.
Zhu, K., & Zhang, T. (2021). Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology, 26*(5), 674–691.