# Reinforcement Learning:Recent Threads

Nimisha Sunny

***Abstract*—Reinforcement learning is a method of training algorithms using reward and punishment feedback. Reinforcement learning agents will interact with their environment to extract information. It is using a trial and error mechanism to learn from its experiences. The goal of reinforcement learning is getting a model that can maximize the total aggregate reward. Its policy is similar to supervised learning. When comparing, both reinforcement learning and supervised learning uses the mapping between input and output as a policy method. This paper contains detailed comparisons and discussion of six reinforcement algorithms, their exploration and exploitation strategy, their weakness and strengths. Background on reinforcement learning models and its recent trends, advantages and future opportunities of reinforcement learning are presented in the paper. This paper is keen to discuss the state-of-the-art applications and achievements of reinforcement learning in various domains.**

***Index Terms*—Reinforcement learning, trends, review, challenges, model-free, Q-learning, DDPG, SARSA, Inverse Reinforcement Learning, Actor critic model.**

## I.INTRODUCTION

Reinforcement learning (RL) is a multidisciplinary machine learning technique. It is a hot research topic in the artificial intelligence domain. The last decade witnessed research and applicational level success in reinforcement learning. The concept and research in reinforcement learning started during 1980 and have its origin rooted back into statistics, game theory, control theory and animal psychology. Reinforcement learning is a goal-oriented method with the ability to interact with the environment and learn from it. A reinforcement learning agent is designed to learn from its environment using a trial and error mechanism. An RL agent in a dynamic environment is capable of extracting information from the current state and to take appropriate action. The reinforcement learning model is designed to choose the best-fit action to maximize the reward and deliver maximum utility. More appropriate actions will be selected based on the reward function value and undesirable actions will be silenced using punishment feedback. Starting from single-agent reinforcement learning detailed research is carrying out on multi-agent and swarm intelligence topics. Genetic algorithms and hybrid approaches are now trying to break the barrier of human cognitive ability to achieve superhuman perfection. The ability of reinforcement learning to extract and manipulate the raw pixels data made it an integral part of computer vision and digital game environments.

An RL agent learns the best policy using the exploration of the environment. Reinforcement learning models are used to solve complex computational problems by setting a reward mechanism and control policies. It chooses a best-fit policy to maximize the reward. The concept of reinforcement learning is inspired and adapted from the natural learning process of the animals, especially the learning and action behaviour of human beings. In reinforcement learning, data will generate based on environment exploration. The agent will only be informed about the starting state, and the behaviour will be modelled based on the reward and punishments. The reward feedback in the form of a scalar objective function is a performance measure of each step. Reinforcement learning is closely related to optimal control theory. RL and optimal control theory are used to find an optimal control policy to optimize the objective function. The ability of RL agents in decision making under the uncertainty maid is special among other machine learning techniques. No information regarding what move to make is provided to the RL agent. The agent must decide the best activity to boost long-term rewards and execute it. The selected action will change the current environment state into the next adjacent state. Different RL algorithms developed during the last two decades are improvised a lot and achieved good results in complex real-world applications. The successful innovations in the topic of RL technologies and algorithms thus demand a comparative critical survey.

This paper discusses the recent trends in RL in terms of application and research level achievements. This review paper is designed to provide a general overview of RL techniques along with elucidating and comparing major reinforcement learning algorithms. This paper is divided into 6 sections. After the introduction, section 2 will give a brief history of RL, a general overview of RL algorithms and classifications and a simple explanation about reinforcement learning model architecture. The core idea of this paper is included in section 3. Here we are providing a detailed analysis and comparison of 6 reinforcement learning algorithms using various classification and performance measurement parameters. The state-of-the-art achievements and applications of RL are provided in section 4. Challenges and future research opportunities in reinforcement learning are discussed in section five.

## II. REINFORCEMENT LEARNING – AN OVERVIEW

### *A. History of Reinforcement learning*

The concept of reinforcement learning originated from 2 different themes. The first one is, trial and error concept developed during 1980. The second method is related to optimal control associated with dynamic programming and value function. The concept of optimal control was introduced in 1950, to minimize the dynamic behaviour of the system. This concept is further developed after the introduction of the "Bell-Man equation". All the methods that use the bell-man equation to solve the optimal control problems are generally called dynamic programming. The markovian decision process, which is known as the base model of reinforcement learning was introduced by Bellman. Paul Werbos proposed heuristic dynamic programming in 1977, which is an approximate approach to dynamic programming. He defined the concepts of dynamic programming and optimal control with computational learning.

The concept of trial and error learning is rooted back to Alexander Bain's concept of "groping & experiment". Edward Thorndike presented the idea of trial and error learning as a learning principle. He defined it as the "Law of effect" because of the reinforcing learn model involved in action selection. The idea of trial and error learning was coined with the concept of artificial intelligence when Alan Turing proposed a design of the pleasure-plain system based on the "Law of effect" (TURING 1996). Richard Sutton who is known as the father of computational reinforcement learning expanded the concept of trial and error learning in reinforcement learning models. He introduced temporal difference learning and policy gradient algorithm, the two most popular fundamental reinforcement learning techniques. The Q-learning algorithm was developed by Chris Watkin by combining the concepts of temporal difference and optimal control.

Several reinforcement learning algorithms and techniques were developed and validated successfully in the last decade. During this period reinforcement learning underwent various improvements and innovations. The next subtopic will discuss reinforcement learning classifications and their evolution along with a brief explanation of the general reinforcement learning model.

### *B. Classification & Model Architecture*

RL models are mainly classified into two types, model-based reinforcement learning, and model-free reinforcement learning. The model is the perception of the agent about its environment. An agent will map states-action pairs to a probability distribution over states. Model-based reinforcement learning methods will choose an optimal policy based on the learned model. In a model-based RL method, the agent has a clear understanding of its environment where it is acting. This environment can be either fully observable or partially observable. Knowing the environment will prepare the agent for choosing the best action and hence maximize future rewards. In a model-based algorithm, the agent will have the capacity to foresee what might happen when picking a specific activity from a scope of potential ones. While model-free reinforcement learning agent is completely unaware of the environment in which it acts. Actions of an agent in a model-free RL model will be limited to a specific date and it doesn't have any idea or knowledge about the next state or outcome of the action. The learning process of model-free RL algorithms is mainly relying upon

experiences. They follow the trial and error learning mechanism for each state-action pair to maximize the rewards to create an optimal policy. Model-free algorithms depend upon instantaneous rewards from a specific state-action pair to evaluate the state utility.
Model-free reinforcement methods are further divided into policy-based (on-policy) and value-based (off-policy) algorithms. The policy-based algorithms will learn the policy without using a value function. The agent will learn the policy function for state-action pairs. In the on-policy models, the policy is defined as π (s, a|θ), where θ is the weight of the input node, s is for state and a is action. The policy-based model will try to optimize the θ using the method of gradient descendent on objective function or maximizing its local assumptions. Policies are of two types, firstly the deterministic policy that is used in deterministic environments like complex s board games. Here policy maps state to activity without any uncertainty. The second method is stochastic policy models which give probability distribution of actions in each state. The value-based reinforcement learning algorithms learn and perform actions without following a policy. The action function θ(s, a) will be decided based on the – how good is an action at a particular state. Commonly these methods will use an objective function defined by the Bell-man equation. In the value-based model, optimization is off-policy which means, the policy used in behaviour generation of training data is independent of estimation policy. Policy gradient algorithms and actor-critic methods are a common example of on-policy, model-free reinforcement learning. The Q-learning, Deep Q-learning, and DDPG are the off-policy methods. Even though DDPG is using both policy and value-based approaches. The model-based reinforcement learning highly emphasizes control function f(s, a). The performance efficiency of model-based methods is defined within a specific environment or in a specific task and it is considered as a limitation of the method. The learning mechanism in model-based reinforcement learning is categorized into two types – learn the model and given the model.
This paper emphasizes different model-free reinforcement learning methods. Major algorithms that come under policy-based and value-based approaches (reinforcement learning, deep reinforcement learning, and Inverse reinforcement learning) are briefly discussed in the below sections .

A simple RL system consists of agents and environment, where the agent will interact and explore the environmental states based on an optimal action. Including agent and environment, a reinforcement learning model consists of six key elements.

• Agent: an agent is the interacting part of the RL system. An agent is a learner as well as a decision-maker.

• Environment: RL agents are designed to interact with the environment and learn from it. The nature of the problem or application defines the environment.

• Actions: Defined as a set of actions, that performed by
the agent. Based on the action new states will be explored.

• States: A state will provide complete information about environment instances, no information is hidden from the state.

• Policy: policy defines the learning and action behaviour of the agent in each state and time. It is the decision-making process and mapping from perceived states to actions. Based on the environment, policies could be stochastic or deterministic.

• Reward signal: A reward is a scalar value and it defines the goal of the RL models. Based on the actions performed the environment will send a reward to the agent. The reward function is defined using actions and states and the equation is given as r=R(s, a).

• Model: The concept of the model is optional, all RL methods are not using a model. A model is the perception of an agent about its environment. The model is used to develop an understanding of the environment.

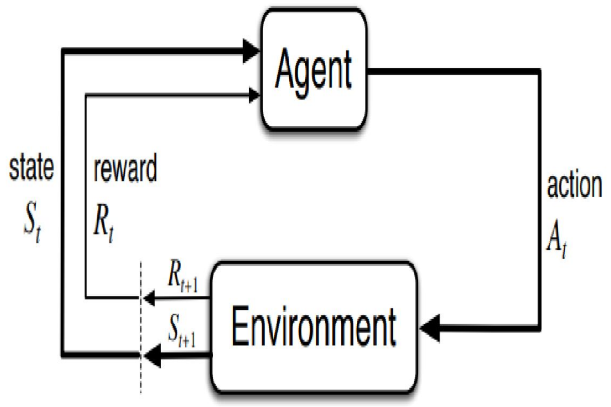

**Fig. 1: Reinforcement learning Model.**

The concept of reinforcement learning is formalized using a Markov Decision Process (MDP). It is very important to discuss MDP because reinforcement learning approaches are following an assumption that it contains an MDP. The Markov Decision Process is used in sequential decision making. In a sequential decision-making model, each action controls and influences the current & next states and the reward of the present action & next action. An MDP is made up of – a set of state (S), set of possible actions (A), rewards (R), and state transition probability (P). In simple words, MDP is a collection of four tuples (S, A, R, P). In a reinforcement learning environment, P and R is unknown. A model-free reinforcement model could only choose a reward based on the trial and error process. So RL models will use a value function to define the long-term reward achievement. Dynamic programming is used to solve problems in an MDP environment where the reward function is present. In an RL model, where the reward function is unknown, here we must use various model-free and model-based algorithms to solve the problem.

A value function defines what is good and ideal for an RL model in terms of long-term reward maximization for a state-action pair. The Monte Carlo method is used for calculating the value function. Monte Carlo method calculates the value of a state by executing many trials runs for a state and find out the final value by taking the average of the trial measurements. Monte Carlo searching is an iterative approximation method and a popular RL algorithm. The temporal difference (TD) algorithm is the basic concept of many RL algorithms. TD was introduced by Sutton, by combining the concepts of dynamic programming with the Monte Carlo approach. TD is the most popular reinforcement learning algorithm. It computes the value function by comparing the temporally successive predictions (Sutton 1988).

## III.REINFORCEMENT LEARNING – ALGORITHMS

### *A. Q-Learning*

The most well known approach algorithm in reinforcement learning is the Q learning algorithm. It is widely used for implementing agents or robots (Kim and Cho 2015). For predicting a future value signal Q learning algorithm is needed. It has the structure of temporal difference learning and also a model-free which makes it an off policy, for instance, each state agent achieves it select a given action on that state and move on to the next state while getting a reward. The purpose of Q learning is to maximize the reward total and with the aid of st = given state, at=action taken, rt=reward, st = next state, +1. Expression of Q learning greedy strategy can be given as:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right]$$

Q learning exploits its environment utilizing the information from present states by action that will surely expand Q[s, a]. It further explores its environment using a greedy strategy in order to get the best Q-function. The above equations demonstrate the total best limit reward of action an agent took in a state according to greedy strategy. The equation has the rate of learning, the factor of discount and condition at initial. Rate of learning decides to what degree recently obtained data abrogate old data, factor discount implies that the future prizes is decided by factor discount and finally the initial factor this stage expects an underlying condition before the initial update with the help of interacting with its environment. Furthermore, exploration can be motivated if the initial has a high number lastly first reward from initial can reset first conditions. The next action of any agent in a particular area is concerned with Q learning to augment the total reward.

Furthermore, the right reinforcement learning algorithm for an assignment such as air combat is Q learning for best action selection. Q learning is recommended due to its method of studying an action-value function that returns a result of a state. State and action of an agent can be gotten while following a pattern using value Q of any action with an unknown model of the environment and can be contrasted with Q learning thereby showing the strength of Q learning over the unknown model. Base plan for the content reserving is in Q learning and it is created to safely store content with collaboration between edge server and that of content supplier. For the shortest path solution Q learning algorithm is the best approach with the aid of the Q table. Furthermore, Q learning calculates value base on the present state and awaiting values. Q table holds the record for Q value in Q learning when an ideal Q table is known the agent begins to venture the area and can choose an ideal activity with the most noteworthy Q esteem in states. N x m is the pattern of a Q table where n=agent actions and m=total number of states. A set of Q values is Q table MBL system can be used to reduce the issues of trial and errors in Q values when a Q table is not available. A system called memory-based learning was utilized to the reduction of trial and error action within a Q value. MBL structure emulates the cerebellum of humans. Furthermore, is a table looking into a method with a table for functions with non-straight values, blocks are utilized for storage of data which can be numerical. Also, blocks are utilized in MBL so as acceleration learning and increment of data can be spread to blocks that are close.

Q learning has its own weak points which affect its performance firstly; time to learn states when it becomes bigger is one of the limitations in Q learning algorithm, due to it learning the process a certain length of time should be available for both action and state. Furthermore locally weighted regression was proposed to tackle one of the major problems with Q learning Algorithm which has to do with it state being recognized as a distinct state. Cubes merger always give intriguing outcomes when algorithms such as Q learning is been actualized on agents with inadequate information or less observation within a domain which through their associations will move together to meet towards a goal (Mourad *et al.* 2014). Q learning is the best approach for the urban environment with the issue of traffic. Regular fixed time traffic sign control normally uncovers low execution when in contact with difficult traffic conditions which is caused by numerous interception with the aid of Q learning model a definite solution we have achieved.

## *B. SARSA*

One of the reinforcement algorithms which has on-policy TD control is Sarsa, its objective approach and conduct policy are both greedy. The major difference between sarsa and Q- learning is that the actual action is taken in sarsa while action with the highest reward is giving more priority and taken first in Q-learning (Jiang *et al.* 2019). Update is done in sarsa with the aid of it five-tuple (s,a,r,s1,a1) which present state is denoted as S, A for action selected in present state after an action the reward gotten is S. a1 and s1 point out the previous state and action when sarsa algorithm gets a reward value back one step which is also referred to as backup the more it gets similar to Q learning. Q value can be updated and learn in sarsa by utilizing the
equation below:

Where present state at a time step t is st, at is the

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

function node selected within a state, rt is the reward, a is the rate of learning (0<a<=1),ϒ has to do with discount rate (0< ϒ <=1) (Park *et al.* n.d.). In sarsa interaction within an environment is necessary before a policy can be updated when an action is made. Sarsa, which is an on-policy method, also utilizes a Q table which includes a matrix with rows and columns which indicate actions and states. Sarsa has various exploration policies like ε greedy and softmax exploration policy. Softmax can be calculated with the aid of Boltzmann distribution. Furthermore, softmax uses probability selection of action which can only be possible by positioning a good estimate of the value function using Boltzmann distribution given by:

$$\pi(a/s) = \frac{e^{\frac{Q^A(s,a)+Q^B(s,a)}{\tau}}}{\sum_b e^{\frac{Q^A(s,b)+Q^B(s,b)}{\tau}}}$$

For a decision on how Q values influence the action is decided by π, greedy action choice is caused as the result of low temperature concerning Q. For all actions to have similar odds of being picked, it uses the result of high temperature. Changing customary strategy like ε greedy, count for state action appearance into the Boltzmann distribution was presented as a strategy for exploration also count based exploration bonus was also included to aid agent exploration during the learning procedure. State to action will be recorded (s,a) count, action (a) can be chosen when an agent is in a state (s): which implies the number of state increase by action of agent in a state finally based count can be utilized in Boltzmann distribution method.

$$prob(a_i) = \frac{e^{count(s,a_i)/T}}{\sum_{a_k \in A} e^{count(s,a_k)/T}} \qquad a_i \in A$$

Prob(a1) indicates the likelihood of executing an action a1, the value of temperature is T, randomness is associated with T so, therefore, the higher the value of T the higher is the randomness. Furthermore, when an agent utilizes the high value of the present value function at the initial stage of learning the issues of exploration and exploitation will arise and the optimal solution may not be achieved because the probability of falling into an optimum which is local is high. For adding exploration bonus to reward the equation is bellowed:

$$R^{+}(s_t,a_t) = \sqrt{\frac{\beta}{\log[count(s_t,a_t)+1]}}$$

β is for bonus and at initial zero is set for all states action (s, a) paired. A reward is denoted R+R+ evaluation for performance is denoted as a total reward without bonuses and finally for count-based sarsa action is not only selected according to the number of times for state action but adding of exploration bonus with respect on a count to the learning procedure. Sarsa equation using count base sarsa is bellowed:

$$\pi(s) = \arg\max_{a \in A} Q(s,a)$$

Value difference-based exploration is utilized in sarsa to minimize unnecessary exploration during episodic tasks due to the unequal exploration of state and this can be accomplished when the information of the initial state has been taken. The main function of VDBE is the inclusion of state-dependent exploration probability as against global parameters. For every time an action is taken in a state it exploration probability will be updated. The equation for VDBE is below:

$$f(s,a,\sigma) =$$

$$\left| \frac{e^{\frac{Q_t(s,a)}{\sigma}}}{e^{\frac{Q_t(s,a)}{\sigma}} + e^{\frac{Q_{t+1}(s,a)}{\sigma}}} - \frac{e^{\frac{Q_{t+1}(s,a)}{\sigma}}}{e^{\frac{Q_t(s,a)}{\sigma}} + e^{\frac{Q_{t+1}(s,a)}{\sigma}}} \right|$$

$$= \frac{1 - e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}}{1 + e^{\frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}},$$

$$\epsilon_{t+1}(s) = \delta * f(s,a,\sigma) + (1-\delta) * \epsilon_t(s).$$

Two parameters are included: σ, $\delta$. When σ has a high value it means large differences are required for exploration. Furthermore lesser changes can make future exploration possible. $\delta$ implies the strength of action which is single on the ε- value of a state. VDBE –Softmax is also an exploration strategy in sarsa that works like the VDBE which implies states. States dependent exploration probability is kept up and assessed when to pick exploration or not and VDBE softmax and this exploration strategy include Softmax behaviour.

$$\pi(s) = \begin{cases} \text{softmax action} & \text{if } \xi < \epsilon(s) \\ \text{argmax}_{a \in A(s)} Q(s,a) & otherwise. \end{cases}$$

Its purpose is to help in situations where few actions produce emphatically negative rewards which when mix with Q value it could be an unnecessary measure of exploration of bad actions. Sarsa as various advantages is a method utilized for issues with negative large rewards furthermore dangerous optima path is always avoided by sarsa during exploration which is also an advantage over other algorithms like Q learning. Sarsa has its own weakness it near-optimal policy during exploration makes it not the best reinforcement algorithm

## *C. Actor critic method*

The actor critic method is on-policy learning. They are also temporal difference methods with entire different memory structures to particularly express the policy independent of a particular value function whereby the policy is called Actor due to its function of chosen actions why the value is the critic. The value is known as a critic because it criticizes actions by an actor; whenever a policy is followed by an actor a critic knows the critique. Actor-critic consists of errors called TD errors which are seen as critique which is output and is the result of both actor and critic.

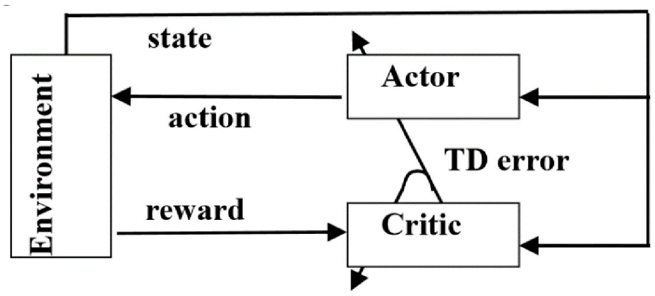

Fig 2: Actor –critic architecture.

Typically, evaluation can only take place after an action has been carried out and a critic is done to know if the outcome is good or terrible. Below is the equation for TD error.

$$\varepsilon_t = r_t + V_t(x_{t+1}) - V_t(x_t)$$

Exploration of action is done by a method called Gibbs softmax which is given as:

P(x, a) is for value at T equal time for actions parameter pointing to the probability of choosing an action (a) when in a particular state. P(x, a) can be updated by the equation below:

$$p(x_t, a_t) = p(x_t, a_t) + \beta\varepsilon_t ,$$

The above equation has a positive β step-size parameter. To have full knowledge about the environment is difficult, before only acting by an agent in a real sense during exploration, exploitation is also considered although there are various popular exploration methods like Ɛ-greedy, Boltzmann and Gibbs softmax which are exploration used in actor-critic. Exploration may be split into two ways firstly for exploration in respect to randomizer which is denoted as an undirected exploration in an environment and exploration for gain in respect to maximization value strategy for gaining which is a direct exploration. Another exploration strategy is the hybrid Gibbs softmax method.

$$\hat{\pi}_t(x,a) = \left(e^{p(x,a)} + \theta/\exp(n_t(a))\right) \Big/ \sum_b \left(e^{p(x,b)} + \theta/\exp(n_t(b))\right) ,$$

Where θ for positive in respect to direct exploration and it can be fixed at a bigger value and reduce water during learning n t (a) equals total times. (a) was utilized for time step t. Policy learning can be improved in the dialogue system using adversarial advantages actor-critic for task completion. (2AC) has popularly known as a policy approach which is to get a policy π which always maximizes reward (R) and tries to minimize j equal to lose. Equation for is the reward and length T while the discount factor. Adversarial advantage actor aim is to encourage the selection of action by the actor which is also guarded by a discriminator in order to enhance exploration. In alternative action selection policies, action like Ɛ greedy is basically in use due to its good exploration techniques. Taking a step to loan and acquire a skill, greedy action towards a goal is not 100% and it also has a minimum percentage of random exploration. For maximum solution issues with random exploration, Boltzmann distribution is utilized

Actor critic method is the combination of actor and critic, actor-critic as various capabilities such as the production of continuous actions, for the evaluation of actor policy critics utilize updating of the value function and the value function is also used to update the actor policy to achieve good performance.A lot of actor-critic is widely used in robotics, power control and finance because of its ability to find an optimal policy when using an estimate gradient of low variance which motivates the speed of learning.Actor critic policy method minimizes variance because TD method can be utilized in finding out critics and exchange variance with bias also other advantages of actor-critic is its gradient update with a perfect critics will allow actor critics to be more example effective through TD update at each progression finally it overpower the issue of high variance gradients in actor only.Actor critic as various Challenges, for instance, applying of actor-critic algorithm on a particular problem even when knowing the experience an actor-critic will not yield a good control policy lastly actor critic algorithm utilizes the same features for actor-critic during experiments which affect its value function.

### *D. Deep-Q Learning.*

Deep-Q learning is a deep reinforcement technique. Deep Reinforcement Learning is derived from the integration of its Learning techniques (DL) with Reinforcement Learning methods(RL). It uses the same principles of DL and RL to generate algorithms that are effective and can be utilized in several sectors such as Video games, Robotics, Finance, and even Healthcare. Implementation of reinforcement learning algorithms with deep learning architecture and deep neural networks will create a powerful model. A deep reinforcement learning framework for learning is to solve sequential decision-making problems. It provides trial and error in the world that provides occasional rewards. DRL is based on training networks like deep neural networks to approximate the optimal policy and/or the optimal value functions V* , Q*, and A*. Policy search methods are mainly focused on two methods gradient-free methods and gradient-based methods. The flow of interest in DRL, avoid the commonly used backpropagation algorithm, which is the gradient-free algorithms.

In deep-Q learning, the agent learns after several times of interaction with its environment. The environment moves to a new state each time an agent chooses an action from a group of certain actions. Punishment and reward are attached to each action as feedback and maximizing the reward will be the next aim of the agent. The formula for Q learning is shown below

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t - Q(s_t, a_t) + \gamma \underbrace{max}_{a'} Q(s_{t+1}, a'))$$

Q(st , at) show the Q value of the agent in a given state st, and time t equal to action at, rewarded with reward rt. The learning rate is α and discount factor equal to γ. utilizing the rate of learning the fastness with which the previous information can be overridden by newly acquired information can be determined. A high value of the learning rate represents a larger change in Q-value. The discount factor function is to determine the usefulness of future rewards. If the discount factor value is closer to 1, then it represents that the future state is more important.

This algorithm is an off-policy control method. It uses updated rules to learn active value function. This method uses the max operator to refine policy-greedily with regards to the action values. This learning is used in spaces that are small where the storage of policy may be in a tabular form. Therefore, a two-dimensional array is used for representing the action space and state space. It uses a dynamic programming strategy to assign values in the array. In this reinforcement learning algorithm, there is a dilemma in choosing between the exploration and exploitation strategy.

On one side, the agent wants to choose maximum possible actions to figure out the optimal strategy, this can be called exploration. On the other side, the agent wanted to select an action with maximum Q-value to increase the reward, this can be called exploitation. As the optimal strategy can only be determined through exploring, exploration strategy is of high importance to learning. But, too much exploration can reduce performance. To overcome this situation, an action selection strategy is used for the learning process where exploration and exploitation are balanced. ε-greedy strategy can resist the system from a local optimal state. It chooses actions randomly with probability $\epsilon \in [0,1]$. Here, the random selection of actions made by the agent in the present state ensures that all state space is explored. By reducing the $\epsilon$ over time, the agent slowly progresses towards exploitation.

### *E.Deep Deterministic Policy-Gradient*

Deep Deterministic Policy Gradient is one of the several Deep Reinforcement Learning algorithms (Tuyen and Chung 2017). This algorithm can deal with high dimensional continuous action space, and it uses Deep Neural Networks (DNN) to represent the policy. The stability and strong learning ability of the algorithm is the reason behind it to be widely used (Pang and Gao 2019). The traditional approaches of Reinforcement Learning (RL) like actor-critic and policy gradient are used by this algorithm. This algorithm, as two main neural networks which are actor-network and critic network. Approximating the policy function is the job of the actor-network whereas approximate the value function is the job of a critic network. The input of a critical network concerns the action with state output from the actor-network. From output, evaluation of the action performed will be made with the help of Q value and the actor updates accordingly in a critical network. Some other methods that are involved in the DDPG method are target network, experience replay, and deterministic policy gradient theorem. The DDPG algorithm framework is given below.

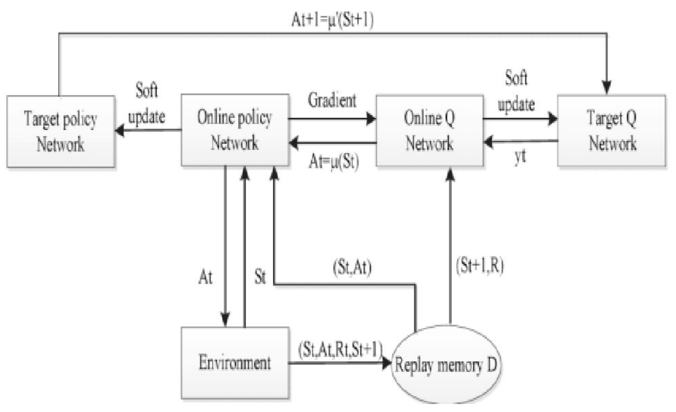

Fig3:Framework Of DDPG Algorithm.

The main advantage of DDPG is that probabilistic representation of the control policy is not needed and hence it is perfect for deterministic policy in several problems. Even though the DDPG method has advantages, it has certain limitations like the actor has to heavily depend on the critic for its learning process, thus the training of DDPG method is sensitive to the efficiency of critical learning.

An off-policy way is used by DDPG in training a deterministic policy. As the policy is deterministic, the agent would not be able to find a wide variety of actions for learning signals at the beginning, if the agent tries to explore on-policy. Noise is added to the actions during training time to make DDPG policies explore better.

To motivate an agent to examine a richer set of state, diversity-driven exploration is an effective approach. With a modification in the loss function, this can be achieved. The modification that has to be made in this case is

$$L_D = L - \mathbb{E}_{\pi' \in \Pi'}[\alpha D(\pi, \pi')],$$

where L represents the loss function of deep reinforcement learning algorithms. The current policy is defined by π, π ' is a policy which is sampled from most recent policies and that set is limited. D is the measure of the distance between π and π '. α is the scaling factor for D. This equation makes an agent proactively try with new policies, which increases the options to visit novel states even if there is an absence of reward signals which is obtained from E. This property is also useful in sparse reward setting, where for most of the states in S, the reward is zero. Also, exploration is motivated by the distance measure D. This is achieved by altering an agent's current policy by π, instead of randomly altering its behaviour. This equation also allows an agent to perform greedy policies while exploring in the training phase. In case of greedy policies, since there is a requirement of an agent for D to update π after the completion of each update, for that state the greedy action may change accordingly, which directs the agent to see unseen states. There are various choices for D, it can be KL-divergence, mean square error (MSE) or L2-norm.

## *F. Inverse reinforcement learning*

An agent is a major component of reinforcement learning (RL) who solves the problems when the agent gets the experience from dynamic environment interactions. Inverse reinforcement learning was introduced due to the issues associated with RL such as design difficulties and advance reward methodology. Inverse reinforcement learning structure in machine learning is developed recently to resolve inverse issues involved in the reinforcement learning algorithm. Inverse reinforcement learning method which can not only be based on the given reward function, but also it gives an observed behaviour from an expert instead of the reward function. Many researchers in various networks like Machine learning, AI, psychology and control theory were attracted by the methods of IRL in the past. IRL engaging a result of its potential to use its recorded data to setup autonomous agents which will have the ability to modelling without the intervention of the performance of the task.

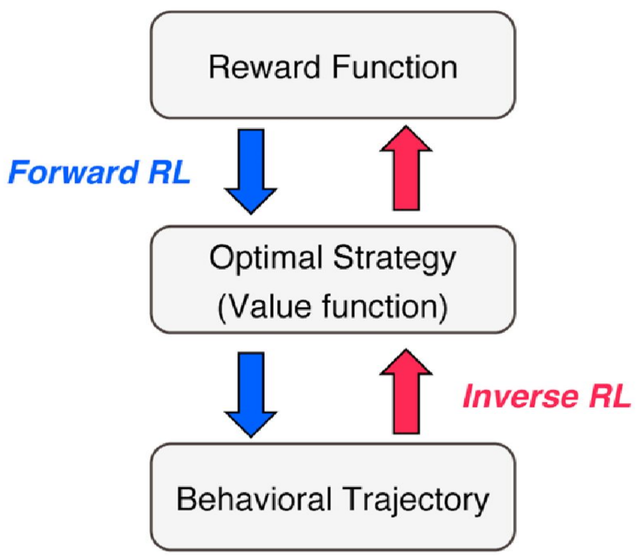

Fig.4: IRL vs RL representation.

There are mainly three types of IRL algorithms("Algorithms for Inverse Reinforcement Learning")

• Firstly, It has Finite-state optimal policy where π is known

• Infinite state optimal policy where π is also known

• Lastly, Infinite state with optimal policy π is unknown

From the above cases of studies, only the last case is close to problems which are Infinite-state with unknown optimal policy(Arora and Doshi 2018).

There are many limitations for IRL, the key problems are related to the award function. For all observation, the behaviour has their reward functions but some of the solutions contain degenerated outputs which are those state's reward value is always zero. IRL depends on the presumption that the expert's policy is ideal concerning an obscure reward function. For this situation, the principal point of the apprentice is to get familiar with a rewarding work that clarifies the observed expert behaviour. At that point, utilizing reinforcement learning, it streamlines its approach as indicated by this reward and ideally carries on just as the expert. Learning a reward has a few points of interest over learning a policy immediately. Firstly, the reward can be investigated to all the more likely to comprehend the expert's behaviour. Secondly, it permits adjusting to perturbations in the dynamics of the environment or it can be transferable to other environments. However, the key problem that is an MDP must solve and obtain reward through optimal policy.

There are mainly three types of IRL algorithms("Algorithms for Inverse Reinforcement Learning")

• Firstly, It has Finite-state optimal policy where π is known

• Infinite state optimal policy where π is also known

• Lastly, Infinite state with optimal policy π is unknown

From the above cases of studies, only the last case is close to problems which are Infinite-state with unknown optimal policy.

There are many limitations for IRL, the key problems are related to the award function. For all observation, the behaviour has their reward functions but some of the solutions contain degenerated outputs which are those state's reward value is always zero. IRL depends on the presumption that the expert's policy is ideal concerning an obscure reward function. For this situation, the principal point of the apprentice is to get familiar with a rewarding work that clarifies the observed expert behaviour. At that point, utilizing reinforcement learning, it streamlines its approach as indicated by this reward and ideally carries on just as the expert. Learning a reward has a few points of interest over learning a policy immediately. Firstly, the reward can be investigated to all the more likely to comprehend the expert's behaviour. Secondly, it permits adjusting to perturbations in the dynamics of the environment or it can be transferable to other environments. However, the key problem that is an MDP must solve and obtain reward through optimal policy.

## IV.APPLICATIONS OF REINFORCEMENT LEARNING

The development of RL is directly related to research in computational game development. Researchers used dynamic programming and RL algorithms to solve complex games (card and board games). In the 21st century, RL is widely used for information retrieval, robotic control and its application is growing beyond the conventional boundaries. Google developed go playing RL agent Alpha-Go and self-taught and playing advanced Alpha-Zero using RL and other hybrid learning approaches. Alpha go confirmed its superhuman proficiency by defeating the world champion Lee Sedol in 2015 in the game of go. In 2017 google developed the most sophisticated board game AI agent Alpha-Zero. The Alpha-Zero was developed using the integration of reinforcement learning with the general-purpose search algorithm. An extension of the policy gradient algorithm named proximal policy optimization was used in the Big-2 game by Charles worth. The counterfactual regret minimization (CFR) was used IIG's and poker AI agent development. CFR is an RL algorithm that only depends upon the number of linear memory data and independent of the number of states. Application of RL algorithms in Atari 2600 games has started the era of RL applications in the interactive digital game domain.

Reinforcement learning approaches are now used for computer resources allocation and scheduling for waiting-tasks. Deep reinforcement learning techniques are used for large scale online multi-resource cluster systems for resource allocation. Traffic signal control is another application
of RL. To solve the traffic conjunction problem, they minimize the block delay by maximizing the reward value using an approximate optimal controller. The developed model is highly scalable and robust. It is appropriate to use in large complicated intersections. Reinforcement learning has an inseparable history with robotics. RL approaches are widely used to improve the behavioural context of robots. RL methods are used in robotics to improve the behaviour learning capability of robots in a dynamic environment. An efficient incremental Q-learning strategy was proposed by Carlucho in 2017, for multi-agent mobile robots. He improved the learning efficiency by defining a time memory as a learning process. The scope of applications of RL for drones and UAVs is wide open and demands further research. UAV is used to accomplish various critical tasks using autonomous coordination and flight strategy. The Q-Learning method is used to develop efficient autopilot and flocking methods. Creating a stabilized autopilot system is an interesting challenge. A neural central model for RL is proposed by Zeng in 2017 to create an efficient flight system. He used this model to send important values into working memory. This model displayed better biological and cognitive properties.

Reinforcement could be used to enhance the transmission control protocol (TCP) in wired and wireless heterogeneous networks. RL based congestion control TCP was developed and used instead of the existing model, which showed high transmission efficiency. Megan and Raj in 2018 proposed an RL based model to control
the loss of separation-based events in the aviation industry to minimize the loss. The proposed model detected the airspace anomalies using the TCAS range tau metric they proposed a safety-event dataset created using self-separation criteria. The application of RL models in biological data deserves special attention. A set of different RL approaches is effectively implemented in various biological data and disease diagnosis methods. Data from medical imaging devices, genetical data and bio-imagining are being classified and analysed and using RL models. These days RL models and algorithms are widely using personalized recommender systems, natural language processing, web and network configuration, computer vision, healthcare management & disease diagnosis and many more.

The application of RL models in various domains is extensive and expensive too. The practical real-world applications of RL have many shortcomings and vulnerabilities. The challenges and weaknesses of the various RL models will be discussed in the next section of the paper.

## V. LIMITATIONS & FUTURE RESEARCH OPPORTUNITIES

### *A. Limitations and Challenges*

Reinforcement learning is used to solve complex tasks across various domains. Its learning mechanism like human cognitive abilities makes it an integral part of AI. Even though RL still has many shortcomings and limitations. The ability of RL agents in their environmental interaction lacks perfection.

Existing exploration and exploitation strategies need improvisation. Exploitation is the process of choosing the best-known policy and action to maximize the global reward, but it doesn't mean it is the best solution in the model. The RL model will follow an exploration strategy to extract more knowledge about the environment and look for better policies to achieve optimal decisions. In a continuous high-dimensional environment, RL algorithms still lack an efficient method of exploration. The most commonly used epsilon-greedy exploration policy treats all actions with the same priority. It's ideal and unguided behaviour is inefficient to identify the promising actions. Reduction in the randomness caused by over-exploitation will lead the policy to trapped inside a local optimum in on-policy algorithms. This will compromise the global objective and leads to the failure of the model. Most importantly, there are no standardized benchmark features to evaluate the performance efficiency of exploration and exploitation strategies.

Lack of real-world training data is another major concern existing in reinforcement learning models. The cost and time of training and learning process are very high, especially for health care and disease diagnosis domain. RL algorithms that need to train from the scratch required large and unbiased training data. Lack of economically feasible advanced computational systems is a bottleneck in real-world applications of reinforcement learning models. Inverse reinforcement learning (IRL) is introduced for autonomous reward function declaration within the RL model. But, the existing IRL model is cable to define the reward function only using the human supported assumptions. The real-world applications of IRL are limited and need more serious research. In real-world applications, the IRL agent has only limited access to the actual environment states. To solve this problem, the IRL model should be integrated with a partially observable Markov decision process (POMDP) .

Applications of reinforcement learning in robotics are still facing severe challenges. The curse of dimensionality and the curse of the real-world are the two general terms used to express the weakness of reinforcement learning in robotics. Curse of dimensionality arises when the size of the solution space of a problem grows exponentially with additional feature exploration from the states. The ability of an RL model in robots to function in a variety of real-world environments is a complex obstacle to tackle. The concept of generalization is a hard task to achieve by RL models. The Open Spiel AI framework released by google achieved a limited generalization ability, but the framework is limited to some specific domains. Along with robust and scalable reinforcement learning algorithms, we need validated evaluation matrices and standardized experimental models to tackle these problems.

### *B. Future Scope and Opportunities*

The future research opportunities in reinforcement learning are bright and wide open. Partial perception problems in a non-Markov environment need detailed research and study. The existing learning algorithms for partial perception problems are defined based on the partially observable MDP, which lacks efficiency. Model-based reinforcement is holding a great future research opportunity. The success of Alpha-Zero proved the efficiency of model-based methods. The reinforcement agents cannot still infer generalized knowledge from different domain-specific tasks and use them in a new environment. RL algorithms still fall behind humans in terms of novel-skill development ability. Domain adaptation-based transfer learning skill ability could be used to learn new experience from a specific task and used it in other.

Application of RL is an

importance in research and application perceptive. The coordination of multiple agents in a heterogeneous environment and creating a stable model is a challenging research task. Further research in the IRL model can develop an efficient method of reward function declaration without any human assumptions. Hybrid reinforcement learning methods, convolutional neural networks for computer vision and hierarchical reinforcement learning for the curse of dimensionality are offering an interesting research opportunity for future researchers.

## VI.CONCLUSION

The last decade witnessed research and application level success in reinforcement learning. The concept of reinforcement learning, which is inspired by animal behaviour is now working on inverse reinforcement learning. Starting with a single-agent RL model, now reinforcement learning is expanding to multi-agent and swarm intelligence.This paper analysed various on-policy and off-policy algorithms like Q-learning, sarsa, Actor-Critic, DDPG, Deep Q-learning, and inverse reinforcement learning based on various classification and comparison features.

The quantitative performance comparison of various reinforcement learning algorithms is limited to specific research and experimental environments. We need validated evaluation matrices and generalized experimental models to provide an accurate cross performance comparison of different algorithms. The curse of dimensionality and the real world are two major existing problems in reinforcement learning. Over the years, improvised RL algorithms could tackle some of its objectives, even though challenges remain, especially in a real-world dynamic environment. RL algorithms still fall behind humans in terms of novel-skill development ability, in the authors' view, the concept of inverse reinforcement learning could overcome this limitation in the near future.

## VII.REFERENCES

Arel, I., Liu, C., Urbanik, T., Kohls, A.G. (2010) ‘Reinforcement learning-based multi-agent system for network traffic signal control’, IET Intelligent Transport Systems, 4(2), 128, available: https://digital-library.theiet.org/content/journals/10.1049/iet-its.2009.0070 [accessed 20 Oct 2019].

Bellman, R. (1957) Dynamic Programming, Princeton University Press: Princeton, NJ, USA.

Busoniu, L., Babushka, R., De Schutter, B. (2006) ‘Multi-Agent Reinforcement Learning: A Survey’, in 2006 9th International Conference on Control, Automation, Robotics and Vision, Presented at the 2006 9th International Conference on Control, Automation, Robotics and Vision, IEEE: Singapore, 1–6, available: http://ieeexplore.ieee.org/document/4150194/ [accessed 20 Oct 2019].

Carlucho, I., De Paula, M., Villar, S.A., Acosta, G.G. (2017) ‘Incremental Q-learning strategy for adaptive PID control of mobile robots’, Expert Systems with Applications, 80, 183–199, available: http://www.sciencedirect.com/science/article/pii/S0957417417301513 [accessed 20 Oct 2019].

Charlesworth, H. (2018) ‘Application of Self-Play Reinforcement Learning to a Four-Player Game of Imperfect Information’, arXiv:1808.10442 [cs, stat], available: http://arxiv.org/abs/1808.10442 [accessed 19 Oct 2019].

Choi, J., Kim, K.-E. (2009) ‘Inverse Reinforcement Learning in Partially Observable Environments’, in Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI’09, Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1028–1033, [accessed 20 Oct 2019].

Dayan, P., Niv, Y. (2008) ‘Reinforcement learning: The Good, The Bad and The Ugly’, Current opinion in neurobiology, 18, 185–96.

Hawley, M., Bharadwaj, R. (2018) ‘Application of reinforcement learning to detect and mitigate airspace loss of separation events’, in 2018 Integrated Communications, Navigation, Surveillance Conference (ICNS), Presented at the 2018 Integrated Communications, Navigation, Surveillance Conference (ICNS), IEEE: Herndon, VA, 4G1-1-4G1-10, available: https://ieeexplore.ieee.org/document/8384897/ [accessed 20 Oct 2019].

Hou, J., Li, H., Hu, J., Zhao, C., Guo, Y., Li, S., Pan, Q. (2017) ‘A review of the applications and hotspots of reinforcement learning’, in 2017 IEEE International Conference on Unmanned Systems (ICUS), Presented at the 2017 IEEE International Conference on Unmanned Systems (ICUS), 506–511.

Huys, Q.J.M., Cruickshank, A., Seriès, P. (2014) ‘Reward-Based Learning, Model-Based and Model-Free’, in Jaeger, D. and Jung, R., eds., Encyclopedia of Computational Neuroscience, Springer New York: New York, NY, 1–10, available: http://link.springer.com/10.1007/978-1-4614-7320-6_674-1 [accessed 19 Oct 2019].

Jansi Rani, S.V., Milton, R.S., Yamini, L., Shivaani, K. (2019) ‘Reinforcement Learning Approach to Improve Transmission Control Protocol’, in 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), Presented at the 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), IEEE: Chennai, India, 1–5, available: https://ieeexplore.ieee.org/document/8862007/ [accessed 20 Oct 2019].

Kalakrishnan, M., Theodorou, E., Schaal, S. (2010) ‘Inverse Reinforcement Learning with PI 2’.

Kober, J., Peters, J. (2014) ‘Reinforcement Learning in Robotics: A Survey’, in Kober, J. and Peters, J., eds., Learning Motor Skills: From Algorithms to Robot Experiments, Springer Tracts in Advanced Robotics, Springer International Publishing: Cham, 9–67, available: https://doi.org/10.1007/978-3-319-03194-1_2 [accessed 18 Oct 2019].

Lanctot, M., Lockhart, E., Lespiau, J.-B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., Hennes, D., Morrill, D., Muller, P., Ewalds, T., Faulkner, R., Kramár, J., De Vylder, B., Saeta, B., Bradbury, J., Ding, D., Borgeaud, S., Lai, M., Schrittwieser, J., Anthony, T., Hughes, E., Danihelka, I., Ryan-Davis, J. (2019) ‘OpenSpiel: A Framework for Reinforcement Learning in Games’, arXiv:1908.09453 [cs], available: http://arxiv.org/abs/1908.09453 [accessed 20 Oct 2019].

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D. (2015) 'Continuous control with deep reinforcement learning', arXiv:1509.02971 [cs, stat], available: http://arxiv.org/abs/1509.02971 [accessed 19 Oct 2019].

Liu, C., Xu, X., Hu, D. (2015) 'Multiobjective Reinforcement
Learning: A Comprehensive Overview', IEEE
Transactions on Systems, Man, and Cybernetics: Systems, 45(3), 385–398.

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, available: https://ieeexplore.ieee.org/document/8277160/ [accessed 20 Oct 2019].

Mao, H., Alizadeh, M., Menache, I., Kandula, S. (2016)
'Resource Management with Deep Reinforcement
Learning', in Proceedings of the 15th ACM Workshop on Hot Topics in Networks - HotNets '16, Presented at the the 15th ACM Workshop, ACM Press: Atlanta, GA, USA, 50–56, available: http://dl.acm.org/citation.cfm?doid=3005745.3005750 [accessed 19 Oct 2019].

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M.,
Bowling, M. (2017) 'DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker', Science, 356(6337), 508–513, available: http://arxiv.org/abs/1701.01724 [accessed 19 Oct 2019].

Ng, A.Y., Russell, S. (2000) 'Algorithms for Inverse Reinforcement Learning', in In Proc. 17th International
Conf. on Machine Learning, Morgan Kaufmann, 663–670.

Nguyen, H., La, H. (2019) 'Review of Deep Reinforcement
Learning for Robot Manipulation', in 2019
Third IEEE International Conference on Robotic Computing (IRC), Presented at the 2019 Third IEEE International Conference on Robotic Computing (IRC), IEEE: Naples, Italy, 590–595, available: https://ieeexplore.ieee.org/document/8675643/ [accessed 19 Oct 2019].

Oh, J., Guo, X., Lee, H., Lewis, R., Singh, S. (2015) 'Action-
Conditional Video Prediction using Deep
Networks in Atari Games', arXiv:1507.08750 [cs], available: http://arxiv.org/abs/1507.08750 [accessed 20 Oct 2019].

Powell, W.B. (2012) 'AI, OR and Control Theory: A Rosetta