A Reinforcement Learning Review: Past Acts, Present Facts and Future Prospects

Benjamin Kommey[1], Oniti Jesutofunmi Isaac[2], Elvis Tamakloe[3], Daniel Opoku[4]
Department of Computer Engineering, Kwame Nkrumah University of Science and Technology[1,2,3,4]
bkommey.coe@knust.edu.gh[1], tofunmi.oniti@gmail.com[2], tamakloe.elvis@gmail.com[3], dopoku-official@yahoo.com[4]

Article Info

Article history:

Received Jun 27, 2023
Revised Aug 8, 2023
Accepted Feb 15, 2024

Keyword:

Reinforcement Learning
Machine Learning
Artificial Intelligence
Deep Learning
Supervised Learning

ABSTRACT

Reinforcement Learning (RL) is fast gaining traction as a major branch of machine learning, its applications have expanded well beyond its typical usage in games. Several subfields of reinforcement learning like deep reinforcement learning and multi-agent reinforcement learning are also expanding rapidly. This paper provides an extensive review on the field from the point of view of Machine Learning (ML). It begins by providing a historical perspective on the field then proceeds to lay a theoretical background on the field. It further discusses core reinforcement learning problems and approaches taken by different subfields before discussing the state of the art in the field. An inexhaustive list of applications of reinforcement learning is provided and their practicability and scalability assessed. The paper concludes by highlighting some open areas or issues in the field.

Corresponding Author:

Benjamin Kommey
Departement of Computer Engineering
Kwame Nkrumah University of Science and Technology
PMB, UPO, Kumasi, Ghana
Email: bkommey.coe@knust.edu.gh

1. INTRODUCTION

As a sub-division of Artificial Intelligence (AI), reinforcement learning has found its relevance in several disciplines such as Computer Science, Engineering, Mathematics, Psychology to mention but a few. Unlike its counterparts, Supervised Learning, and Unsupervised Learning, it operates by placing an agent in an environment, the agent takes an action, and then the interaction is translated into a reward, as well as an associated state. The primary goal of the agent is to magnify the reward despite the initial uncertainty of the environment. Its versatility has made it applicable in game theory, control systems, robotic control, multi-agent systems and statistics. One of the oldest and perhaps, profound, uses is in solving control system problems using the most optimal route. Another remarkable application is that of the AlphaZero, also referred to as the “first multi-skilled AI board-game champion”. Notably reinforcement learning is amongst the paradigms in AI. The others are Supervised and Unsupervised learning. With supervised learning, a classified and labelled data set is used. The model is trained using the data to predict the output for a different input. In supervised learning system, performance improves by increasing the number of training examples. Typical supervised learning problems consist of object detection, image captioning, classification, regression, and labelling [1]. Though this learning type is relatively simple, it is not a perfect- fit for interactive environments. In such environments, learning would be more efficient if the system could learn from its own experience [2]. This leads to unsupervised learning. Unsupervised learning operates on unlabeled datasets, and it strives to find a pattern from such datasets. It aims at analyzing and deducing the underlying structure of the data. Unsupervised learning problems include feature learning, clustering, dimensionality reduction, and density approximation or estimation [1]. The third concept is reinforcement learning (RL). This category of learning entails the interaction of the agent with the environment. The agent acts on, and it receives feedback or reward. It is a trial-and-error method in which the agent experiences both failures and successes while trying to maximize the reward [1]. The unique feature has been verified to be a modern method to build a human-level agent [3]. Each time step, the agent perceives the current state of the environment and then takes an action. Every action causes the agent to transit to a new state. For each transition, there is a reward signal that determines the magnitude of the transition. In case of a wrong action [failure] the agent receives a negative transition and for a right action [success], it receives a positive transition. Ultimately, the agent is tasked with maximizing the cumulative reward [4].

Alternatively, the focus of RL is to strike and find a midpoint between exploration and exploitation [5]. All three types of learning are encompassed under Machine Learning and Artificial Intelligence. RL differs from supervised learning because it requires no labelled dataset, and the reward feedback is less informative in RL than in supervised learning where the agent would be given the correct actions to take [6]. Though RL shares some common attributes with unsupervised learning, there are differences between them. RL focuses on maximizing the reward rather than finding the hidden structure [2]. Also, the feedback in RL is much more informative when compared with unsupervised learning, no explicit feedback on the performance can be found [7]. The advent of Deep Reinforcement Learning has stirred up a huge technological advancement and companies such as Google, Uber and Tesla are incorporating it into self-driving autonomous cars.

This paper gives a comprehensive survey on the state of the art of reinforcement learning. It explores, discusses, compares, and critiques the various state of the art techniques, methods of reinforcement learning as well as its applications. Limitations are discussed in each case and future research directions are proposed. This paper serves as a good starting point for researchers looking to conduct research on any branch of reinforcement learning. In this regard, the remainder of the document is structured as follows: Section 2 talks about the contribution and related works. In Section 3, the background to reinforcement learning is well explained. Section 4 dwells on the applications of RL. Section 5 presents the results and discussion, and this contribution wraps up with the conclusion in Section 6.

2. CONTRIBUTIONS AND RELATED WORKS

There exist several papers in the literature that have surveyed either the entire field or a subfield of reinforcement learning. This section discusses the contributions of those papers, their approach, and their shortcomings which this work aims to cover. Kaelbling et al [8] provides the first comprehensive survey of reinforcement learning. Their work detaily discusses some of the core issues in reinforcement learning like exploration and exploitation, generalization, and hierarchy, and dealing with hidden states. It further reviews and assesses the practicability of some implemented systems. While this is a detailed review, the field has evolved quite a lot since this work was published in 1996. In [9], Agostinelli et al briefly review RL, considering its history, the state of the art and place emphasis on first principles. Benjamin Retch [10] provides a survey of reinforcement learning, but unlike [8], the viewpoint is that of optimization and control and the focus is presenting its applications in continuous control. Fengji et al [11] summarize advances in model-based RL, presenting characteristics, advantages, and defects of each approach as well as their applications across several fields. Considering how wide the field of reinforcement learning is, several authors have decided to survey specific subfields. In [12] Arulkumaran et al, provide a brief survey on deep reinforcement learning. The paper reviews various value-based and policy-based methods, it covers the main algorithms present in deep reinforcement learning such as deep Q-networks (DQN), asynchronous actor critic, and trust region policy optimization (TRPO). It details several open areas of research within the field as well. As this is intended to be a brief survey, it fails to mention works that provide contributions to the field. It also fails to provide current areas of application of deep RL. Furthermore, quite a lot has changed in deep RL since this paper was published in 2017. Mousavi et al [13] improves on [12] by focusing on current developments in deep reinforcement learning. It discusses some of the more frequently used architectures like recurrent neural networks (RNNs), convolutional neural networks (CNNs), and autoencoders.

Just like [12] however, it is brief and leaves out several significant works and does not discuss applications. Yuxi Li [14] surveys recent advancements in deep reinforcement learning. It provides a background to machine learning, deep learning, and reinforcement learning. It further discusses core reinforcement learning problems like policy, reward, planning and models. It discusses 12 applications of reinforcement learning including games, natural language processing (NLP), finance, text generation, industry, and a lot more. While this is the most comprehensive work on deep reinforcement learning to date, it fails to sufficiently discuss open issues in the field. Mohammad et al [15] surveyed the application of Bayesian methods in reinforcement learning. It covers extensively recent advances on both Bayesian methods for model-based and model-free reinforcement learning. In summary it provides a survey on Bayesian Reinforcement Learning algorithms, alongside their theoretical and empirical properties. Busoniu et al [16] is an overview of the field of approximate RL. It comparatively analyzes different methods and techniques for approximate RL. Multiagent reinforcement learning (MARL) systems are finding increasing number of applications across a wide variety of domains. There are a few papers in the literature dedicated to surveying this specific subfield of reinforcement learning. Busoniu et al [17] comprehensively surveyed the field, providing a detailed description of RL MARL algorithms. It also discusses some domains of application for MARL and provides a map of the field. A lot of significant contributions have been made in the field since its publication. [18] is another somewhat similar version of [17]. Hernandez-Leal et al [19] surveyed multi-agent deep reinforcement learning (MDRL). The work emphasizes the practical challenges of MDRL. One major difficulty in MARL is the coordination of multiple agents to reach the required goal.

Choi et al [20] provided a survey on the different coordination strategies for multiple agents including cooperation and competition. The applications of reinforcement learning are indeed varied, and several papers have set out to survey the applications of RL in specific domains. Polydoros et al [22] surveyed the application of model-based reinforcement learning to robotics with an emphasis on algorithms and hardware. Kober et al [21] provided a general survey of the application of RL in robotics, considering both model-free and model-based methods. It considers the successes and challenges of applying RL in robotics. It also does a great job outlining open issues and possible research areas. In [23] Shao et al provided a survey on the application of deep reinforcement learning to video games. It reviews the application of RL to a variety of video games including 2D and 3D games. Additionally, it considers important topics to consider when designing RL systems for games such as: sample efficiency, exploration-exploitation, generalization and transfer, multi-agent learning, incomplete information, and delayed spare rewards. Mahmud et al [24] discussed the application of deep learning and reinforcement learning to biological data. Yu et al [25] provided a survey on the application of reinforcement learning in healthcare.

Multi Agent Reinforcement Learning (MARL) which involves multiple agents performing similar tasks learn faster by sharing experience and exchanging communication. The ability of other agents to take over some of the tasks if one or more agents fail makes it robust [4]. However, existing MARL algorithms require some preconditions before the benefits above can be exploited [28, 29]. Game theory which involves the theory of learning in games [30] has been linked with MARL. Researchers such as Shoham et al. [31] have delved into the connection between MARL and game theory using some of the MARL algorithms to back up their claims. Engineering design is a highly iterative problem-solving process [32]. Design automation presents a solution by transferring the tasks from a human designer to the computer. Previous research on this topic was limited to the use of Bezier curves to represent designs [33]. However, Fabian Dworschak et al. presented a general method for design automation by applying RL in parametric Computer Aided Design (CAD) models.

The parametric CAD Models to depict the design as learning environment for the RL agents expanded the scope. RL could be used to predict designs that are manufacturable since restrictions can be implemented in the parameter grid. Furthermore, they showcased the concept of transfer learning for related design tasks in a bid to ensure automation [34].

Robot design is an extremely arduous and complex task considering that there are many parameters that could alter its final performance. Legged robots even pose a bigger challenge as literature on design principles is very sparse [35]. It is often unclear as to how the final values are determined [36]. To combat the design optimization problem, Álvaro Belmonte-Baeza, Joonho Lee et al. suggested a quantitative model-free approach based on Meta Reinforcement Learning. They introduced an adaptive RL-based locomotion controller, whose locomotion policy is conditioned on the design parameters such that it can optimize each design instance. The Meta RL ensures fast adaptation of the policy to the specific design and on application to the leg link length of four-legged robots, they obtained a policy that resulted in close-to-optimal locomotion control of the robot. Their framework provided a substantial improvement and generated lower-cost designs [35].

Aside from research of RL in robotics and game theory, Abedali El Gourari et al. also delved into the implementation of Deep RL in e-learning and distance learning [26]. They adopted a framework for promoting learning and focused on the implementation of Deep Q-Networks to train the agent or a virtual teacher to do Remote Practical Work (RPW) in a short period. Then the agent can help improve the cognitive levels of the students by informing them of things they must do and things they do not need to do to understand the subject. In summary, the agent uses Deep RL to learn how to take the perfect actions to solve the problem in its environment. After training, the interactive RPW platform is provided for the student and the student is then evaluated by the virtual teacher [37].

Table 1. A comparison with related works

Related works	Specified domain	Merits and Demrits	Algorithm description	Discussed limitations	Future prospects	Performance comparison
Kaelbling et.al.[8]	✓	✓	✓	✓	×	×
Recht et.al.[10]	✓	×	✓	✓	×	×
Fengji et.al.[11]	✓	×	✓	×	×	×
Arulkumaran et.al.[12]	×	✓	✓	✓	×	×
Li et.al.[14]	✓	✓	✓	×	×	×
Buşoniu et.al.[16]	×	✓	✓	×	×	✓
Buşoniu et.al.[17]	✓	✓	✓	✓	✓	×
Choi et.al.[20]	×	×	✓	✓	✓	×
Kober et.al.[21]	✓	✓	✓	✓	✓	×
Shao et.al.[23]	✓	✓	✓	✓	×	✓
Mahmud et.al.[24]	✓	×	✓	✓	✓	✓
Yu et.al.[25]	✓	✓	✓	✓	✓	×
Gourari et.al.[26]	✓	✓	✓	×	×	✓
Price et.al.[29]	×	✓	✓	×	×	×
Shoham et.al.[31]	×	✓	✓	×	×	×
Viquerat et.al.[33]	×	✓	✓	×	×	×
Dworschak et.al.[34]	✓	✓	✓	×	✓	✓
B-Baeza et.al.[35]	✓	✓	✓	×	×	✓
This work	✓	✓	✓	✓	✓	✓

To highlight the contributions of this work, a comparison with the related works is presented in Table.1. In this table, the specified domain shows whether the reviewed works captured the application or targeted areas. Furthermore, the implications of the solution (whether positive or negative) as expressed or discussed by authors are indicated by the merits and demerits. Algorithm description is a mark given to denote that related works presented the theoretical or mathematical concept behind reinforcement learning models and solutions. Moreover, if the authors mentioned and elaborated on the open issues and limitations with regards to their work, it is represented in the discussed limitations column. On the other hand, future prospects indicate if the paper detailed the expectations that are likely to resolve the current issues or challenges encountered. Additionally, there exists a final indicator to show whether the performance assessment of reinforcement learning methods are good, accurate, efficient, and reliable for application in real life environments based on distinct or unique metrics.

2. RESEARCH METHOD

The emergence of Reinforcement learning has seen its application in several areas such as robotics, manufacturing, communication, education, image processing, healthcare, and entertainment among others. In this regard, many significant gains have been made through their utilization in these respective areas. For instance, in image processing, reinforcement learning has positvely influenced the extraction of vital information from images, enhancing image quality for analysis in computer vision related tasks in healthcare and many more. Similarly, a notable success in reinforcement learning is its application in the gaming arena. That is, in learning from games played by professionals or specialist (Alpha Go), stochastic games, learning by self-play (Shogi and Chess) as well as from learning without employing hand-crafted elements or features (Atari games). In reference to motion and its dynamics, reinforcement learning has been employed in several control tasks such as balancing of a pendulum without initial knowledge of its operating dynamics, performing low speed hovering with a helicopter in an inverted manner and other defined maneuvers. It is therefore imperative to note that through reinforcement learning, solutions have been provided to numerous challenges in a very optimized and efficient manner.

2.1. Main concept

As a sub-field of machine learning (ML), reinforcement learning employs software agents that interact with its immediate surroundings to enhance performance by achieving the most desirable cumulative reward. Along with supervised and unsupervised learning techniques, reinforcement learning (RL) is another integral machine learning paradigm that is feedback-based and with either positive or negative learning classification. In typical RL problems, a decision maker is made to operate in an environment modeled by states $s_t \in S$. The decision maker alternatively referred to as the agent can take certain operational actions or measures at $\in A(s_t)$ as a function of the recent (current) state s_t. Following the selection of an action at time t, the agent acquires a scalar reward $r_{t+1} \in R$ and subsequent arrives in a new state s_{t+1} based on the recent state and chosen action. It also follows the policy π_t strategy, at every time step. This implies a mapping from states to the likelihood of choosing each possible action: where π (s, a) represents the likelihood that $a = a_t$ if $s = s_t$. Thus, the goal of reinforcement learning is to utilize the interactions that agents have with their immediate environment to obtain (derive) the most advantageous policy to magnify the agent's gross reward received over time.

2.2. Definitions

Model

A model is the term referring to the different varying states of the environment and the resultant reward associated with each state. It merely makes it possible to draw conclusions and make predictions about the environment. Using a model is optional in reinforcement learning thus leading to model-based (dynamic programming) and model-free reinforcement learning. The distinction between the two is the use of models. In the model-based approach, the agent is enabled to construct a functional representation of its environment (model) from which the agent uses to plan and decide the next course of action. It deviates from the "trial-and-error". The model comprises transition probabilities and the reward function [4, 38, 39]. Conversely, the model-free method forgoes understanding the environment to make decisions but rather, the agent acts given a particular state. No model or planning is involved. The agent freely learns on the go and a change in the environment or action of the agent is only possible after exploration [40].

According to Sutton and Barto who defined RL as learning to behave optimally in a stochastic environment by performing actions and getting rewards, model-based methods depend on planning as their fundamental component, while model-free methods basically depend on learning [2]. They analogized it to the habitual and goal-oriented control of learned behaviour patterns. However, Sutton claimed the model-based is the next revolutionary stage of AI. It promotes safe exploration and effective exploration though achieving such feat is quite a formidable task [41].

Return

To achieve maximum cumulative reward in the long-term after the current time (t), with regards to a finite time horizon that ends at time T, the return R_t is equal to:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T = \sum_{k=t+1}^{T} r_k \qquad (1)$$

In the case of an infinite time horizon, it is customary instead to use a discounted return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots += \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \qquad (2)$$

By assuming that the rewards are bounded and $\gamma < 1$, convergence is bound to occur. As a result, $\gamma \in [0, 1]$ is a constant. This notation is referred to as the discount factor. This discounted definition for the return would generally be employed in what follows.

Agents

This refers to the entity undergoing testing and training to make effective and correct decisions. It could be a four-legged robot being trained to move, a robot arm learning to play Chess. The agent plays an important role and through its perception and actions rewards are earned. The aim of the agent is to pick the optimal policy that maximizes the long-term reward but if on weighing the agent's current performance to the optimal performance there is a lag that births the concept of *regret* [45]. Additionally, there can be a single agent or multi-agents undergoing the training. With the single agent, it is modelled by the Markov Decision Process (MDP) where one agent has all those characteristics, while in the multi-agent scenario, there is a generalization of the MDP where several autonomous, cooperating units share the same environment, which they identify or recognize with sensors and act upon with the aid of actuators [46-48]. It represents the stochastic game.

In the stochastic game, there is a tuple $(S, A_1, \ldots, An, R, P_1, \ldots, P_n)$, consisting of the n number of agents, where S is still the environment states, $A_1, \ldots, A_n$ is the set of actions available to the agents, R is the reward of the agents assumed to be bounded and $P_1, \ldots, P_n$, are the probability transition functions. Multi-Agent systems are used in Multi-Agent Reinforcement Learning (MARL). They can be applied in a vast number of fields like distributed control [49], resource management [50], robotic teams [51], etc. [27, 52]. Apart from the robust nature, it also provides a high degree of scalability; new agents can be easily inserted into the system. The agents can take advantage of parallel computation and experience sharing, through communication [53], for higher speed [54] and better performance respectively [55]. Despite the huge perks associated with MARL, it is limited by the curse of dimensionality, stability, exploration-exploitation tradeoff and nonstationarity [1].

Environment

Agents are placed in a particular surrounding or space in which they have no control over except the actions they take in the environment. The environment consists of different states and the decision made determines the current state. A simple analogy is that of a robot placed in a maze, the robot has no sway over the obstacles, it can only solve the maze if the right series of actions are made.

Value functions

This is the long-term value of a state or an action. It is the predicted **return** over a state or an action making it crucial to evaluate states in S and selecting action. To obtain optimal behavior or policy, one would think the straightforward approach is to list all the possible polices and pick the policy with the highest possible value for each initial state. That approach is not realistic, hence value function. Value functions are calculated first, such that the optimal value $V^*(s)$, $s \in S$ yields the highest return. The expression below represents the optimal value function [7].

$$V * : S \rightarrow R$$

Return (G_t) is the long-term total discounted reward. Rewards are short-term gains. A high reward value from one state to another does not infer that the action made produces the greater total reward in the long run.

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

Value functions can be categorized into state-value functions and action-value functions. State-value function V(s) is the expected return when starting from a state, *s* [2].

$$V_s = E[G_t | S_t = s$$

where t is the timestamp.

Action-value function $q(s, a)$is the expected return starting from state, **s**, taking an action *a*, with respect to a policy π. The mathematical formula is as below: [40]

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a$$

For an optimal policy to be found, certain algorithms are based on value functions, V (s), which indicates how profitable it is for an agent to arrive at a given state s. This function offers, for every state, a numerical approximation of the potential future reward achieveable from this state, and is therefore followed by the agent based on the original policy π:

$$V^\pi(s) = E_\pi[R_t | s_t = s] = E + \pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right] \qquad (3)$$

where Eπ[.] indicates the expected value provided the agent follows policy π, and t is any time step.

Action-Value functions

Here, the action-value function Q is stated as the value of taking an action a in state s under a policy π:

$$\begin{aligned} Q^\pi(s, a) &= E_\pi[R_t | s_t = s, a_t = a] \\ &= E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right] \end{aligned} \qquad (4)$$

Optimal policy

Formally, the policy is defined as a mapping between the state set and the action set. $\pi: S \rightarrow A$ For an agent to make an action, there is a thought process involved and that is defined by the policy. Each action has a probability distribution associated with it and highly rewarding actions tend to have a high probability. The policy determines the behavior of an agent, and it is the possible actions that the agent should take for every possible state, s ∈S. A policy can either be stochastic (nonzero probability of multiple actions selected) or deterministic (predetermined mapping of states to actions) [44]. This can either be a lookup- table, a function, or a search process. The core of RL is to find the optimal policy.

A policy that attains the most expected reward over the long run is termed as optimal policy $\pi *$. Hence, provided its expected return is more than or equal to the expected return for all states, a policy π is better than or equal to a policy π*. Therefore:

$$\pi^* = argmaxV^{\pi}(s) \ \ \forall s \in S \qquad (5)$$

Markov Decision Processes (MDP)

To better understand reinforcement learning, a sound knowledge of Markov Decision Process is crucial. The environment in which the agent is located, is in MDP form because many RL algorithms use the model-free techniques [42-43]. It is a mathematical framework for formulating a discrete-time decision making process [35]. In reinforcement learning, an MDP is expressed as a distinct case in which the set of states and actions of each state are finite. The Markov characteristics are therefore satisfied under these stated conditions:

$$P_r\left(s_{t+1} = s^{`} \middle| s_0, a_0, \dots, s_t, a_t\right) = \ P_r(s_{t+1} = s^{`} | s_t, a_t) \qquad (6)$$

This implies that the likelihood of arriving in state **s** from state **s** by action a is discrete of the other previous states or actions prior to a set time (t). Thus, successive states, actions, and rewards gathered from a Markov Decision Process can be denoted or represented by a decision network. Majority of reinforcement learning research are established on the decorum of Markov Decision Processes. MDPs offer basic frameworks in which to study simple algorithms and their associated features. The MDP is a 5-tuple in the form (S, A, P, R, γ) where S is the finite set of the States, A is the set of actions of the agent, P is the state in the transition probability function from a state, $\mathbf{S_i}$ to $\mathbf{S_j}$ (where i to j is the timestep) under a particular action a_i, R is the reward function determining the immediate reward the agent receives on transition to the next state, and γ is the discount factor which caters for the uncertainty of future rewards. The above terminologies will be explained below.

- The first element **S** is the state space. It refers to all the possible internal states of the agent. Each state is just the position of the agent in the environment.
- Next is the action space, **A**, which contains the actions of the agents. The action refers to the choice the agent makes at the current time step.
- State transition probability function or transition probability function has two possible values, [0, 1]. If it is 1, this means the transition is possible from $s_{i,}$ to s_j when the agent performs an action a_i. For a value of 0, it is not possible.
- Lastly, R, which is the reward function whose output determines the reward associated with completing an action – either good or bad. Depending on the reward value, the policy is updated.

In summary, at every timestep, the agent receives a state, $s_{i,}$ and under a particular action a_i, there is a probability to transition to, s_j, as determined by $P\left(s\middle|i, a_i, s_j\right)$. The reward received is a function of $R(s_i, a_i s|j)$ [4]. What characterizes a Markov Decision is that the probability of the next state is dependent on the present state and not on the past states, meaning the future is independent of the past given the present [1]. The goal of MDP is to get the optimal policy that maximizes the utility or cumulative discounted reward of the agent.

Exploitation and Exploration

The major distinction between RL and Supervised Learning is the issue of exploration. Only by exploration can the agent better understand the environment. It is required and important for the agent to explore. However, the cost of exploration, but in most cases, conflicts with exploitation. This tradeoff presents one of the most predominant issues in RL. A classic example is the case of the multi-armed bandit scenario in [45]. Exploitation involves utilizing the agent's current knowledge and exploitation is information-gathering actions taken to improve that knowledge. In MARL, this is a challenge because of multiple agents. The agents can explore independently to obtain information about the environment and other agents to adapt [4]. However, exploring too much can disorient the other agents leading to increasing in the learning complexity for the agent exploring. The eventual goal is to achieve safe and effective exploration while balancing exploitation [56].

2.3. Algorithmic approaches

For a typical reinforcement learning problem, the optimal policy can be computed using different methods or approches. Primarily, there are two methods: that is, the search in the space of value fuctions and the search in space of policies. The search in the space of value fuctions seeks to determine V* which denotes the optimal value function and derive by inference the optimal policy π^* from V* to conclude. This method consists of Monte-Carlo, temporal difference, linear and dynamic programming methods. Conversely, policy space search keeps explicit description of policies and improve or update them over time to determine the optimal policy π^*. Typical techniques include evolutionary and policy gradient algorithms. An overview of these methods is presented in the following sections.

Algorithms are finite sequences or steps used in solving a problem. RL is about achieving an optimal policy to inadvertently maximize the return. To do this, there are various methods, each with their strengths and weaknesses. This section will touch on the famous Q-learning, SARSA, Monte Carlo and Temporal Difference.

Linear programming

To find the optimal value function in reference to linear programming problems, V which represents the value function is treated as a cost function. Afterwards, a measure to minimize the cost from each starting state **s** is adopted. In this case, it is achieved by inverting the sign of the rewards. The cost function is noted by $g(s_t) = r_{t+1.}$ Hence, the need to minimize:

$$J^{\pi}(s) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k g_{\pi}(s_k) | s_0 = s\right] \quad (7)$$

To carry out the minimization, it is important to define the optimal Bellman operator T:

$$(TJ)(s) = \min\,(g_{\pi}(s) + \gamma P_{\pi}(s) J \quad (8)$$

here, J denotes a vector of states, P_π represents the transition matrix with the (s, s) entry depicting the probability of arriving at **s** from **s** under policy π, and the minimization is performed component-wise. Therefore, the solution that achieves minimization of the cost should verify the Bellman equation:

$$J(s) = (TJ)(s) \quad (9)$$

It can be determined by computing the linear programming optimization (for instance, employing the simplex algorithm):

$$\begin{aligned} &min_J \quad \mu^T J \\ &\text{s.t.} \quad \text{TJ} \geq J \end{aligned} \qquad (10)$$

where μ denotes a vector of positive weights, referred to as the state-relevance weights. Theoretically, linear programming presents the only known algorithm capable of solving MDPs in polynomial time, even though their approaches to reinforcement learning problems generally do not thrive or perform well in practice. Particularly, the major problem with regards to linear programming approaches is that the space and time complexities can be immensely high.

Dynamic programming

The simplest or effortless method to address a reinforcement learning problem is by employing dynamic programming algorithms. Nonetheless, this method needs a detailed understanding of the model under consideration and is limited by the computational cost. The concept behind the dynamic programming formulation of reinforcement learning is to select a policy, approximate its cost value function V (Algorithm 1), derive from V (Algorithm 2) a new policy, and iterate this very process until a suitable policy is found (Algorithm 3). This technique is mostly referred to as policy iteration. This is because the policy is strictly improved in each step, and the algorithm is certainly guaranteed to converge with the optimal policy. In relation to computational convenience, one can opt to terminate or halt the policy evaluation step when the change in the value function is very minimal (between two iterations), as performed below with the threshold.

Q-Learning

Q-learning was introduced by Chris Watkins in 1989 in his Ph. D thesis [39]. The idea behind Q-learning which makes it a simple reinforcement algorithm is that it stores data in tables. It is a model-free RL method that learns long-term optimal behavior by finding mappings from state-action pairs to values. Those values, also known as Q-values, are calculated from the reward function. The Q-value obtained can help in improving the estimated solution [40].

$$\Delta Q(s_t, a_t) = \alpha\delta = \alpha(R_t + \gamma Q(s_{t+1},\ a) - Q(s_t,\ a_t)) \qquad (11)$$

This refers to the expected discounted reinforcement of taking an action in state and refers to the learning rate and temporal difference error. Q-learning is an off-policy, that is, it evaluates the target policy while following a different policy (behavior policy). To find the optimal action value function, a greedy approach is used. The Q-table is initialized. The agent then performs an action. On completion, a reward is obtained and used to update the Q-table using the formula below.

$$Q^{new}(s_t\ , a_t) = Q(s_t, a_t) + \alpha\delta \qquad (12)$$

The result is to obtain Q-values that converge to the optimal values, and the greedy policy applied by the agent should converge at the optimal policy as the learning rate improves. However, its weakness is that, as the state space gradually increases, defining a Q-table would be an arduous task. Deep Q-Network (DQN) presents a solution by using neural networks with Q-learning to derive approximate Q-values in the same fashion as the standard Q-learning.

SARSA

This algorithm is a variation of the Q-learning technique which stands for State Action Reward State Action. Unlike Q-learning, it is an on-policy method, meaning that the target policy is the same as the behavior policy; the agent uses a single policy. Off policy methods tend to produce better policies than on policy but as the environment complexity increases, it experiences scalability concerns [2]. Also, on-policy methods have better performance. SARSA, initially called Modified Connectionist Q-Learning (MCQ-L), was introduced by Rummery and Niranjan [41].

The concept is simple. The updated Q-value depends on the current state, current action, reward received, the next state, and the next action.

$$Q^{new}(s_t, a_t) = \alpha(R_t + Q(s_{t+1}, \; a_{t+1}) - Q(s_t, \; a_t)) \qquad (13)$$

Monte Carlo and Temporal Difference

With Monte Carlo (MC) method, the agent learns directly from the episodes. An episode is the length of a simulated learning process starting at a selected state and it ends at a terminal state. It is also a model-free approach. The rewards obtained during the episode are sampled and the average is taken to give the MC return. Given ample time, MC provides a precise estimate of the optimal policy. However, MC has some limitations including that it only works in episodic problems and that it can only be applied in small, finite MDPs because of the possibility of delay when processing a suboptimal policy.

Temporal Difference (TD) does not wait for the outcome of each episode but learns from incomplete episodes. It utilizes sampling, as well as a form of bootstrapping. Bootstrapping, in the sense that, the returns are adjusted to be more precise before the episodes end. TD solves majority of the limitations attached to MC. It also has the value that lies between 0 and 1. To get the best of both MC and TD, they can be altered [60].

2.4. Performance metrics

To assess the reliability of RL algorithms, research by [61] proposed a standardized measure of performance. They suggested 7 metrics to provide reproducibility or stability both during training and rollout. These metrics catered for both measures of variability, dispersion, and risk.

1. **Dispersion across Time (DT)**
   This metric occurs during training. The algorithm must display smooth monotonic improvement rather than noisy fluctuations across time. In a bid to ensure that short-term changes should be emphasized and not long-term trends (detrending).
2. **Short-term Risk across Time (SRT)**
   The most extreme short-term drop over time is calculated to determine the worst-case drop in performance during training. Higher drop indicates an unstable algorithm.
3. **Long-term Risk across Time (LRT)**
   This determines if an algorithm has the potential to lose performance relative to its peak. It measures unusually large drops that occur over longer timescales.
4. **Dispersion across Runs (DR)**
   RL algorithms should have reproducible performance across multiple training runs. Previous research has discussed this metric [62-64]. Inter Quartile Range (IQR) or variance or standard deviation can be calculated to ensure that the algorithm remains consistent.
5. **Risk across Runs (RR)**
   Measuring this helps give the anticipated results of the bad runs. It is applied to the results of the entire training runs.
6. **Dispersion across Fixed Policy Rollouts (DF)**
   Fixed policy is obtained after training. It determines the variability in results when the same policy is rolled out multiple times.

7. **Risk across Fixed Policy Rollouts (RF)**
   RF shows the expected loss in the worst-case scenario when the same policy is applied on rollout performances.

The above-mentioned metrics reveal the strengths and weaknesses of an algorithm. Other performance metrics to consider are win rate or success rate, median performance, cumulative reward over time, performance percentile, regret, time to converge, episode length [65]. Apart from the reliability, another important aspect to consider is the stability of the learning process by the agent. Some metrics to help to evaluate include the statistical variance of the cumulative reward, sensitivity to initialization and sensitivity to noise.

Abdelali El Gourari et al. conducted research into the use of Deep Reinforcement Learning in E-learning and Distance Learning. It entails an agent aiding, guiding, and interacting with the student in their remote practical work. Though RL has achieved great feats in relation with decision making and complex problem solving, however the algorithms used perform better with good data. For this reason, they used Deep Q-Networks (DQN) which is a union of deep learning and RL. DQN uses neural networks to approximate the Q-value. After training the model thoroughly, the results were promising and eventually it yielded a 94% accuracy. A similar work by El Fouki et al [66] presented an accuracy of 99% using the same DQN algorithm. Agrebi et. al [67] obtained 48% accuracy with DQN and lastly Shahbazi and Byun [68] used Double Deep Q-Network which produced 21.46% accuracy [26]. In Robotics, Alvaro et al. implemented Meta RL for the optimal design of legged robots. Meta RL was used to ensure fast adaptation to the policy for a specific design during optimization. The meta-policy used enabled quick adaptation and an optimized design in about 1.4 h after 72 hours of training [35].

2.5. Reeinforcement learning applications

It goes without saying that the versatility and the prowess of RL has made it applicable in diverse fields of humanity ranging from psychology to medicine. RL is employed by tech giants like Google, Uber, and Tesla in autonomous driving (self-driving cars). Other applications include route optimization, trading and finance, healthcare, intelligent transportation systems, marketing, and games.

Robotics

RL has been used to train robots to perform a variety of tasks such as navigation, performing surgeries, grasping objects and so on. It has found its way from even the design up to the control of robots. Alvaro Belmonte-Baeza et al delved into obtaining the optimal design of legged robots. Robot design is a complicated and intricate process that depends on several parameters. [69] describes such parameters like gear ratio, limb lengths and so on. What makes the design process cumbersome is because of the huge involvement of human intellect in the design. The ideal way of countering this baffling situation is to incorporate computational design methods which would thus optimize the design parameters. However, current designs are rigid; hence not flexible to present a more holistic solution. The panacea suggested in their research was to use meta-RL to develop a locomotion policy which quickly adapts to different designs [70]. Such a policy would provide an unprecedented solution hinged on versatility of robotic designs [35]. Another interesting application is the work of Scheal and Atkeson. They constructed a two-armed robot that juggles a devil-stick after hundreds of attempts. The policy used by the robot was continually improved using a form of dynamic programming. [4, 61-63]. Mahedevan and Connell designed a mobile robot to perform the task of *box-pushing,* which happens to be a tedious task for robots to perform. With the use of Q-learning (an RL algorithm) and some clustering techniques, the robot performed quite competitively with a robot pre-programmed with a human solution. [4, 74].

Autonomous driving

With the advent of AI (Deep Learning), autonomous systems such as self-driving cars or drones have been brought to light. Such systems can attain an impressive level of awareness using Deep Learning, but they still need to optimize the decisions made to ensure ultimate performance and this is what RL does. In reference to autonomous driving, this is essentially a multi-agent setting whereby the host vehicle applies complex negotiation skills by taking left and right turns whiles moving ahead in a nebulous and unorganized urban road. The two major challenges with RL for Autonomous Driving are that of the unpredictability of other drivers in traffic and the functional safety of the driving policy (the long-term driving strategy of the car) [75]. Deep Reinforcement Learning has been applied in autonomous systems. RL enables the systems to learn from the data obtained from the environment to make the correct decision. RL algorithms are used for the decision making and maneuver execution systems like lane change and keeping [76-81], overtaking maneuvers [82], intersection and roundabout handling [83-84]. According to [76], there are two crucial components of autonomous driving systems - planning and control systems. The planning systems predict the path the self-driving car should take while the control systems are responsible for low-level actions like controlling steering angles, throttle, and break.

Paolo Maramotti et al. researched in developing a system that unifies both components into a single module using the delayed version of the Asynchronous Advantage Actor Critic (A3C) algorithm [85-87]. They pre-trained the model using Imitation Learning (IL) to resolve the issue of requiring large number of episodes for training. At the end of both simulation and real-world testing (using *deep_response* module) in obstacle-free environments and low-traffic neighborhood respectively, the agents performed well. The planner developed was able to predict acceleration and steering angle every 100ms [75]. Though the experiment was limited by the size of visual inputs, the simulated agents were able to evolve conveniently.

Trading and finance

Another important and timely application of RL is in the trading and finance industry. Here agents are trained to automate trading by incorporating both AI and RL [70]. The process of trading has been parted into market condition summarization and optimal action execution [89]. What makes it exceptionally challenging is the dynamic decision making based on a set of frequently changing factors. [89] investigated the use of RL and DL in a complex neural network. The DL senses trends and changes in the market conditions while the RL ultimately makes the decisions to cumulate the rewards. The model was tested on real-world financial data, and it yielded a high level of accuracy. Reinforcement learning has been adopted to provide solutions to several financial issues such as investment and portfolio allocation, hedging and pricing contingent claims, market making, asset liability management, buying and selling a portfolio of securities subject to transaction costs, and optimization of tax consequences [90]. The assumptions made in the classical approaches are avoided in RL because it learns to optimize the gain regardless of the cost function provided. Hence, making it potent in financial applications.

FinRL, an open-source framework, has been developed as a massive aid to traders in quantitative finance. It implements DRL (Deep Reinforcement Learning) algorithms to simulate a wide array of markets as well as trading constraints to decide where to trade, at what price and what quantity [91]. DRL addresses the dynamic decision-making problems by offering portfolio scalability and market model independence [92-96]. This gives it a competitive edge over human traders [97-98]. Due to the versatility and simplicity of FinRL, it can be applied in stock trading [93], portfolio allocation and cryptocurrencies trading [99]. It comprises of three layers, application layer, agent layer and environment layer. All these layers are fully customizable. In summary, FinRL aids in developing a DRL trading strategy by overcoming the error-prone programming and strenuous debugging phase.

Games

This is without doubt the most common application of RL. Games like checkers [100], tic-tac-toe [101], backgammon [102], Go [103] consists of a series of alternating moves, thus applying RL is not an improbable task [104]. Thanks to RL, computers have surpassed human levels of performance in Chess, Checkers, Othello, Backgammon, Scrabble and so on. To achieve this, a straightforward strategy is applied. [105] breaks the process in three-folds. Though the above-mentioned games differ from each other, the goal is to first analyze the positions. In Chess, the pawn structure, King safety is evaluated [106]. For Checkers, materials and degrees of mobility are assessed [107]. As for Scrabble, the single, duplicate, and triplicate letters are analyzed [108]. After analyzing the positions, the agents are trained using TD-learning and self-play to fine-tune their performance. Finally, a search algorithm is applied such as Minmax in the case of Chess, Checkers and Othello or Monte-Carlo for Scrabble [108].

[105] delved into how RL can be applied to the game of Go which seems to be a challenging problem for most computer programs to master [109]. However, DeepMind were able to succeed in training AlphaGo program using RL to defeat the Go's world champion, Lee Sedol. Another notable achievement is AlphaZero that mastered both Chess, Shogi and Go within 24 hours of self-play by reusing the same hyperparameters for the three games [110]. Games are a perfect application for RL because the agent can explore its virtual environment at an affordable cost.

2.6. Route optimization and path planning

The process of ascertaining the most cost-efficient route is widely referred to as Route Optimization. It is more than just identifying the shortest path between two points as there are many factors and complexities to consider. Autonomous and semi-autonomous systems like unmanned ships make use of route optimization and path planning for maritime transportation, reconnaissance, and intelligence training [111]. Unmanned ships face unique problems due to the degree of harshness of the environment. Strong waves, current surges and so on can affect their navigation capabilities [112-113]. [111] probes into the local path planning of mobile unmanned ships to ensure optimal paths of the unmanned ships using Deep Reinforcement Learning. Initially, unmanned ships have no knowledge of their environment during the exploration. Thus, they are prone to the old-fashioned exploration vs exploitation dilemma.

Di Wi et al. used the path corner waiting for optimization to shorten the total travel time of the unmanned ship crossing a path. The unmanned boat used the deep policy gradient methods to optimize the policy and improve the reward function. The training sessions were set to 5000 [114-115]. It resets to a new one when the boat collides with an obstacle. After 1000 sessions, the success rate increased and so the average cumulative reward increased. [106] combined two RL algorithms, Q-Learning and SARSA to compare and find the optimal path for mobile robots. The algorithms were tweaked to make learning faster. Globally Guided Reinforcement Learning (G2RL) was employed in [117] for large dynamic environments, where the obstacles move. The mobile robots used G2RL, a hierarchical path planning approach, to navigate the environment. The experiment also was conducted on multi-robot path planning which yielded a successful result [118].

3. RESULTS AND ANALYSIS

RL has a wide array of uses both unexplored and explored. Though some challenging problems have been solved, some questions are still unanswered. Discerning the appropriate RL algorithm to apply can be quite challenging. They all have their respective strengths and weaknesses. Some give improved performances over others depending on the applications. Alvaro et al, in the field of robotics, incorporated Meta-RL in the design optimization of legged robots. The outcome was a flexible, dynamic and an optimized design that could adapt to changes in the design instance – including unexpected changes. The framework was lightweight and low-cost compared to other frameworks and the nominal designs. The catch, however, was the realistic cost functions of the framework as they could not account for the inner workings of the system such as the power consumption and transmission losses [43].

[4] researched into the benefits of MARL over single agent reinforcement learning. Theoretically, MARL poses a lot of prospects because it involves multiple agents coalescing together to solve a problem by learning from each other. Practically, MARL is applied to small problems due to concerns of scalability. Only a subset of MARL algorithms can perform considerably with limited or incomplete knowledge. However, with further research, challenging realistic world problems can be solved using MARL.

Deep Reinforcement Learning has proved its relevance in diverse applications of RL. It is a blend of Deep Learning and Reinforcement Learning. The DQN algorithm, a model free RL algorithm, links deep neural networks and RL. Google's DeepMind made use of it to create AlphaGo, [119] applied it in stock market predictions and self-driving cars implement it in their operations.

Ergo, [26, 66-68] exploited DQN in their research into using RL in e-learning and distance learning. [77] compared DQN to other algorithms such as LinUCB, HLinUCB, W&D. DQN had the highest accuracy of 48%. Table 2 shows the various accuracy of different researchers using DQN or DDQN algorithms.

Table 2. The Performance of various algorithms

Researchers	Algorithms	Accuracy
Abdelali El Gourari et al. [26]	DQN	94%
El Fouki et al. [48]	DQN	99%
Agrebi et al. [49]	DQN	48%
Shahbazi and Byun [50]	DDQN	21.46%

In view of this, the complex problems across different domains require very impactful solutions that further enhance or improve the capability, reliability, efficiency, and applicability of reinforcement learning. Thus, a concise description of the prevalent areas in RL is provided in Table 3.

Table 3. Recent trends in reinforcemment learning

Scope	Descirption
Deep Reinforcement Learning Balhara et.al.[120]	This allows for the combination of deep neural networks with RL algorithms to harness their combined potential. Hence, this facilitates a more complex but scalable learning in high dimensional spaces with technques such as deep deterministic policy gradient (DDPG), DQN, proximal policy optimization (PPO) and DDQN. Thus, neural networks are currently employed in the extraction of high dimensional observation features, estimating Q values of states, and increasing robustness in deep reinforcement learning models.
Multi-Agent Reinforcement Learning Li et.al.[121]	MARL focuses on systems where many agents interact, learn, collaborate, and compete effectively. Thus, current methods are exploring the decentralized approach for agents operating in complex and adverse environments while ensuring scalability. Improvements are made equally to algorithms whiles maintaining the conventional network structures and eliminating centralization. In this manner, coordination between multiple agents is enhanced.
Transfer learning and Meta-Learning Upadhyay et.al.[122]	In the quest to increase reliability, RL models must learn, and leverage knowledge or experience gained in one environment to accelerate execution of tasks in similar environments. Since the different learning paradigms individually have merits and demerits, a hypothesized merging of transfer, meta and multi-task learning are under consideration to create generic learning networks in RL. Hence, model adaptability is a key area making remarkable strides
Sample Efficiency and Exploration Strategies Yang et.al.[123]	This involves improving RL algorithms to depend less on many interactions with the environment to learn effectively whiles ensuring maximum efficiency. In this regard, curiosity-driven, meta-explorations and episodic memory modules among other techniques have been developed to improve sample efficient reinforcement learning architectures. Based on this, significant gains have been made with the incorporation of episodic thoughts into other vital DRL modules such as loss function and experience relpay.
Safety Reinforcement Learning Jeong et.al.[124]	This entails the continuous process of guaranteed operation of RL agents within predefined safety boundaries to avoid undesirable results. Thus, methods for learning policies that reduce the chances

	of unsafe or destructive actions are increasingly becoming extremely essential in areas like robotics, umanned driving and the financial market. A typical case is the development of the novel safety asset allocation reinforment learning (AARL) framework. This operates by combining multiple protective dynamic asset allocation strategies (PDAS) to reduce risks via RL.

4. OPEN ISSUES AND LIMITATIONS

The challeneges or limitations of RL, although undesired, has opened numerous possibilities of solutions proposed to address them. Inspite of the novel architectures, algorithms, reward structures and exploring hybrid techniques, there exist some key areas or aspects that still require immense attention to tackle the limitations. The current limitations in the field of RL are as follows:

Sample efficieny challenges

To effectively learn optimal or near-optimal policies and perform well, RL agents often require a substantial number of interactions with the environment. This is computationally expensive to realize thus limiting its applicability in real-world scenarios where data might be scarce and capital intensive to acquire [125]. Practically, sample efficiency challenges are linked to exploration and exploration where balacing trade-off without wasting interactions on suboptimal actions are difficult to achieve. Additionally, the high-dimensional state spaces, sparse rewards and sample complexity increases the challenge of finding the optimal strategy, receiving feedback results, and achieving convergence.

Generalization and TransferLearning issues

Generalizing learned policies to new, unseen environments or tasks remains a huge problem. Evidently, RL models often struggle to transfer knowledge efficiently across different domains or adapt to changes in the environment. Furthermore, for RL agents that employ functional approximation such as neural networks to generalize, it is difficult to ensure that the approximations generalize well across several states without either overfitting or underfitting [126]. Moreover, catastrophic forgetting exists where there are no considerations for well established techniques for continuous learning and experience replay.

Safety, Stability and Ethical concerns

Ensuring safety and ethical behaviour of RL agents in complex, real-world environments is critical. However, RL algorithms may learn behaviours that are unsafe or undersirable and hence make it essential to develop methods to enforce safety constraints and ethical behaviours related to biases and fairness. In adversarial environments, RL agents are vulnerable to manipulation and exploitations by adversaries leading to unsafe sates. Convergence to optimal policies has also been a struggle for many RL algorithms resulting in erratic and instability in performance or behaviours [127]. More so, due to the lack of interpretability in RL models, explaining its decision-making processes are laborious. This raises ethical concerns with regards to societal impact when deployed.

Exploration and Exploitation Tradeoff

Balancing exploration, which involves trying new actions to discover better strategies with exploitation which leverages known strategies for immediate reward, is crucial. Many RL algorithms are faced with the challenge of efficiently exploring the environment while maximizing rewards, especially in complex environments having sparse rewards. Thus, more advanced exploration techniques are needed for this effect [128, 129]. Additionally, exploring can be risky or less rewarding in the short term, especially in situations whereby the actions chosen by the RL agent does not promptly show or yield high rewards. On the other hand, using exploitation in the short term although reliable may result in RL agents missing potentially superior actions or strategies. Hence, finding the right balance via trade-off can prove difficult.

Sample Complexity in High Dimensional Spaces:

RL algorithms encounter stiff difficulties in high-dimensional state or action spaces where exploration becomes more problematic, and learning can be ineffective because of the curse of dimensionality. Others work in the bid to maintain low sample complexity experience challenges with achieving high predictive power even with model-based methods [130]. The significant increase in sample complexity when dealing with high dimensional action spaces offers huge restrictions to the RL agent with regards to exploration. Thus, with the lack of coverage across the states, effective and comprehensive learning is impractical to realize in such cases.

5. CONCLUSION

This paper sought to provide a solid basis to reinforcement learning by highlighting the various building blocks that compose it. It discusses the various types of RL and how they have been applied in different fields of humanity. We aimed at revisiting the roots of RL, its status, and future prospects. To advance the prowess of RL, the major limitation, computational capability must be addressed. A tremendous boost in computational power leads to increased throughputs. More efficient algorithms should also be developed. Further research in RL should be centered on unlocking the full potential of RL to make great strides in decision making and tackle complicated tasks.

ACKNOWLEDGEMENTS

Authors would like to acknowledge and appreciate Hermann Kumbong formerly of Kwame Nkrumah University of Science and Technology, Department of Computer Engineering for the initial support of the work.

REFERENCES

[1]. J. T. Sri Suma A. Hammoudeh, "A concise introduction to reinforcement learning," Princess Suamaya University for Technology: Amman, Jordan 2018.

[2]. R. Sutton, A. G. Barto, 2nd ed. in progress, and M. The MIT Press: Cambridge, USA, "Chapter 1 The Reinforcement Learning Problem," pp. 1-25, 2017.

[3]. L. Tai and M. Liu, "Towards cognitive exploration through deep reinforcement learning for mobile robots," 2016.

[4]. L. Buşoniu, Lucian, Robert Babuška, and B. De Schutter, “Multi-agent reinforcement learning: An overview,” Chapter 7 in *Innovations in Multi-Agent Systems and Applications – 1 (D. Srinivasan and L.C. Jain, eds.),* vol. 310 of Studies in Computational Intelligence, Berlin, Germany: Springer, pp. 183–221, 2010.

[5]. V. Cherkassky and F. Mulier, Learning from Data Concepts, theory, and methods. New York: J. Wiley, 1998.

[6]. G. Hinton and T. J. Sejnowski, Unsupervised learning: foundations of neural computation. MIT press, 1999.

[7]. L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[8]. Kaelbling L. P., Littman M. L., Moore A. W. Reinforcement Learning: A Survey, 1996, J. Artif. Intell. Res.

[9]. Agostinelli F., Hocquet G., Singh S., Baldi P. (2018) From Reinforcement Learning to Deep Reinforcement Learning: An Overview. In: Rozonoer L., Mirkin B., Muchnik I. (eds) Braverman Readings in Machine Learning. Key Ideas from Inception to Current State. Lecture Notes in Computer Science, vol 11100. Springer, Cham.

[10]. Benjamin Recht, A Tour of Reinforcement Learning: The View from Continuous Control (2019). Annu. Rev. Control Robot. Auton. Syst. 2019.2:253–79

[11]. Yi, Fengji; Fu, Wenlong; and Liang, Huan,” Model-based reinforcement learning: A survey” (2018). ICEB 2018 Proceedings. 60. https://aisel.aisnet.org/iceb2018/60

[12]. K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath," Deep Reinforcement Learning: A Brief Survey," in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[13]. Mousavi, Sajad & Schukat, Michael & Howley, Enda. (2018). Deep Reinforcement Learning: An Overview. Lecture Notes in Networks and Systems. 426-440. 10.1007/978-3-319-56991-8 32.

[14]. Yuxi Li, Deep Reinforcement Learning: An Overview. CORR, abs/1701.07274 2017.

[15]. Ghavamzadeh, Mohammed et al. "Convex Optimization: Algorithms and Complexity." Foundations and Trends in Machine Learning 8.5-6 (2015): 359–483. Crossref. Web.

[16]. L. Bu¸soniu, D. Ernst, B. De Schutter and R. Babuˇska," Approximate reinforcement learning: An overview," 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), Paris, 2011, pp. 1-8, doi: 10.1109/ADPRL.2011.5967353.

[17]. L. Busoniu, R. Babuska and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 38, no. 2, pp. 156-172, March 2008, doi: 10.1109/TSMCC.2007.913919.

[18]. Bu¸soniu L., Babuˇska R., De Schutter B. (2010) Multi-agent Reinforcement Learning: An Overview. In: Srinivasan D., Jain L.C. (eds) Innovations in Multi-Agent Systems and Applications - 1. Studies in Computational Intelligence, vol 310. Springer, Berlin, Heidelberg

[19]. Hernandez-Leal, P., Kartal, B. & Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. Auton Agent Multi-Agent Syst 33, 750–797 (2019). https://doi.org/10.1007/s10458- 019-09421-1

[20]. Y. Choi and H. Ahn," A survey on multi-agent reinforcement learning: Coordination problems," Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, Qingdao, 2010, pp. 81-86, doi: 10.1109/MESA.2010.5552089.

[21]. Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 32(11), 1238–1274. https://doi.org/10.1177/0278364913495721

[22]. Polydoros, A.S., Nalpantidis, L. Survey of Model-Based Reinforcement Learning: Applications on Robotics. J Intell Robot Syst 86, 153–173 (2017). https://doi.org/10.1007/s10846-017-0468-y

[23]. Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li and Dongbin Zhao. A Survey of Deep Reinforcement Learning in Video Games, (2019).

[24]. M. Mahmud, M. S. Kaiser, A. Hussain and S. Vassanelli," Applications of Deep Learning and Reinforcement Learning to Biological Data," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 6, pp. 2063-2079, June 2018, doi: 10.1109/TNNLS.2018.2790388.

[25]. Chao Yu and Jiming Liu and Shamim Nemati, Reinforcement Learning in Healthcare: A Survey (2019).

[26]. A. El Gourari, M. Raoufi, M. Skouri, and F. Ouatik, "The implementation of deep reinforcement learning in e-learning and distance learning: Remote practical work," *Mobile Information Systems*, vol. 2021, pp. 1–11, 2021.

[27]. A. L. C. Bazzan and F. Klügl, "Introduction to Intelligent Systems in Traffic and Transportation", vol. 7, no. 3. 2013

[28]. M. Lauer, "An algorithm for distributed reinforcement learning in cooperative multiagent systems." In *Proc. 17th International Conf. on Machine Learning*. 2000.

[29]. B. Price and C. Boutilier, "Accelerating reinforcement learning through implicit imitation," *Journal of Artificial Intelligence Research*, vol. 19, pp. 569–629, 2003.

[30]. D. Fudenberg and D. K. Levine, *The theory of learning in games*. Cambridge, Mass.: MIT Press, 1998.

[31]. Y. Shoham, R. Powers, and T. Grenager, “If multi-agent learning is the answer, what is the question?,” *Artificial Intelligence*, vol. 171, no. 7, pp. 365–377, 2007.

[32]. G. Pahl, W. Beitz, Engineering Design: A Systematic Approach, Springer Science & Business Media, 2013.

[33]. J. Viquerat, J. Rabault, A. Kuhnle, H. Ghraieb, A. Larcher, and E. Hachem, “Direct shape optimization through deep reinforcement learning,” *Journal of Computational Physics*, vol. 428, p. 110080, 2021.

[34]. F. Dworschak, S. Dietze, M. Wittmann, B. Schleich, and S. Wartzack, “Reinforcement learning for engineering design automation,” *Advanced Engineering Informatics*, vol. 52, p. 101612, 2022.

[35]. A. Belmonte-Baeza, J. Lee, G. Valsecchi, and M. Hutter, “Meta reinforcement learning for optimal design of Legged Robots,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12134–12141, 2022.

[36]. C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, “Design of hyq – a hydraulically and electrically actuated quadruped robot,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[37]. F. Ouatik, M. Raoufi, M. El Mohadab, F. Ouatik, B. Bouikhalene, and M. Skouri, “Modeling collaborative practical work processes in an e-learning context of engineering electric education,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, no. 3, p. 1464, 2019.

[38]. A. W. Moore and C. G. Atkeson, “Prioritized sweeping: Reinforcement learning with less data and less time,” *Machine Learning*, vol. 13, no. 1, pp. 103–130, 1993.

[39]. R. S. Sutton, “Integrated Architectures for learning, planning, and reacting based on approximating dynamic programming,” Machine Learning Proceedings 1990, pp. 216–224, 1990.

[40]. D. Silver, “Deep reinforcement learning,” in International Conference on Machine Learning (ICML), 2016.

[41]. N. D. Daw, S. J. Gershman, B. Seymour, P. Dayan, and R. J. Dolan, “Model-based influences on humans' choices and striatal prediction errors,” *Neuron*, vol. 69, no. 6, pp. 1204–1215, 2011.

[42]. B. H. Abed-alguni, S. K. Chalup, F. A. Henskens, and D. J. Paul, “A multi-agent cooperative reinforcement learning model using a hierarchy of consultants, tutors and workers,” *Vietnam Journal of Computer Science*, vol. 2, no. 4, pp. 213–226, 2015.

[43]. M. Van Otterlo and M. Wiering, "Reinforcement learning and Markov decision processes," pp. 3-42, 2012.

[44]. D. Silver, R. S. Sutton, and M. Müller, "Reinforcement Learning of Local Shape in the Game of Go," in IJCAI, 2007, vol. 7, pp. 1053-1058.

[45]. D. A. Berry, B. J. L. C. Fristedt, and Hall, "Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)," vol. 5, no. 71-87, pp. 7-7, 1985.

[46]. Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[47]. N. Vlassis, "A concise introduction to multiagent systems and distributed artificial intelligence," vol. 1, no. 1, pp. 1-71, 2007.

[48]. G. Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[49]. H. V. D. Parunak, "Industrial and practical applications of DAI," pp. 377-421, 1999.

[50]. G. Tesauro, J. O. Kephart, "Pricing in agent economies using multi-agent Q-learning," vol. 5, pp. 289-304, 2002.

[51]. P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," vol. 8, pp. 345-383, 2000.

[52]. B. Bakker, M. Steingrover, R. Schouten, E. Nijhuis, and L. Kester, "Cooperative multi-agent reinforcement learning of traffic lights," 2005.

[53]. M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330-337.

[54]. L. Busoniu, B. De Schutter, and R. Babuska, "Multiagent Reinforcement Learning with Adaptive State Focus," in *BNAIC*, 2005, pp. 35-42: Citeseer.

[55]. C. Guestrin, M. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *ICML*, 2002, vol. 2, pp. 227-234: Citeseer.

[56]. P. Osinenko, D. Dobriborsci, and W. J. I.-P. Aumer, "Reinforcement learning with guarantees: a review," vol. 55, no. 15, pp. 123-128, 2022.

[57]. C. J. C.H. Watkins, "Learning from delayed rewards." (1989).

[58]. C.J.C.H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[59]. G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[60]. G. Tesauro, "Temporal difference learning and TD-Gammon," vol. 38, no. 3, pp. 58-68, 1995.

[61]. S. C. Chan, S. Fishman, J. Canny, A. Korattikara, and S. Guadarrama, "Measuring the reliability of reinforcement learning algorithms," 2019.

[62]. Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*, 2016, pp. 1329-1338: PMLR.

[63]. M. Fortunato *et al.*, "Noisy networks for exploration," 2017.

[64]. M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International conference on machine learning*, 2017, pp. 449-458: PMLR.

[65]. S. Jordan, Y. Chandak, D. Cohen, M. Zhang, and P. Thomas, "Evaluating the performance of reinforcement learning algorithms," in *International Conference on Machine Learning*, 2020, pp. 4962-4973: PMLR.

[66]. M. El Fouki, N. Aknin, and K. E. El. Kadiri, "Intelligent adapted e-learning system based on deep reinforcement learning," in *Proceedings of the 2nd International Conference on Computing and Wireless Communication Systems*, 2017, pp. 1-6

[67]. M. Agrebi, M. Sendi, and M. Abed, "Deep reinforcement learning for personalized recommendation of distance learning," in *New Knowledge in Information Systems and Technologies: Volume 2*, 2019, pp. 597-606: Springer.

[68]. Z. Shahbazi and Y. C. Byun, "Toward social media content recommendation integrated with data science and machine learning approach for E-learners," vol. 12, no. 11, p. 1798, 2020.

[69]. M. Chadwick, H. Kolvenbach, F. Dubois, H. F. Lau, M. Hutter, "Vitruvio: An open-source leg design optimization toolbox for walking robots," vol. 5, no. 4, pp. 6318-6325, 2020.

[70]. C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *international conference on machine learning*, 2017, pp. 1126-1135: PMLR.

[71]. W. S. Cleveland and S. Devlin, "Locally weighted regression: an approach to regression analysis by local fitting," vol. 83, no. 403, pp. 596-610, 1988.

[72]. A. W. Moore and C. G. Atkeson, "An investigation of memory-based function approximators for learning control," Tech. rep., MIT Arti cal Intelligence Laboratory, Cambridge, MA1992.

[73]. A. Sage and C. C. White, "Optimum systems control. Prentice Hall," 1977.

[74]. S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," vol. 55, no. 2-3, pp. 311-365, 1992.

[75]. S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016.

[76]. P. Maramotti, A. P. Capasso, G. Bacchiani, and A. Broggi, "Tackling Real-World Autonomous Driving using Deep Reinforcement Learning," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1274-1281: IEEE.

[77]. P. Wang, C. Y. Chan, and A. De La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1379-1384: IEEE.

[78]. C.J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning.," presented at the 21st International Conference, Intelligent Transportation Systems (ITSC), 2018

[79]. B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2156-2162: IEEE.

[80]. A. E. Sallab, M. Abdou, E. Perot, and S. J. a. p. a. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," 2016.

[81]. A. Feher, S. Aradi, and T. Becsi, "Q-learning based reinforcement learning approach for lane keeping," in 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), 2018, pp. 000031-000036: IEEE.

[82]. M. Kaushik, V. Prasad, K. M. Krishna, and B. Ravindran, "Overtaking maneuvers in simulated highway driving using deep reinforcement learning," in *2018 IEEE intelligent vehicles symposium (iv)*, 2018, pp. 1885-1890: IEEE.

[83]. L. García Cuenca, E. Puertas, J. Fernandez Andrés, and N. Aliane, "Autonomous driving in roundabout maneuvers using reinforcement learning with Q-learning," vol. 8, no. 12, p. 1536, 2019

[84]. A. P. Capasso, G. Bacchiani, and D. Molinari, "Intelligent roundabout insertion using deep reinforcement learning," 2020.

[85]. V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928-1937: PMLR.

[86]. A. P. Capasso, G. Bacchiani, and A. Broggi, "From simulation to real world maneuver execution using deep reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1570-1575: IEEE.

[87]. A. P. Capasso, P. Maramotti, A. Dell'Eva, and A. Broggi, "End-to-end intersection handling using multi-agent deep reinforcement learning," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 443-450: IEEE.

[88]. E. W. Saad, D. V. Prokhorov, and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," vol. 9, no. 6, pp. 1456-1470, 1998.

[89]. Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, and l. systems, "Deep direct reinforcement learning for financial signal representation and trading," vol. 28, no. 3, pp. 653-664, 2016.

[90]. P. N. Kolm and G. Ritter, "Modern perspectives on reinforcement learning in finance," vol. 1, no. 1, 2020.

[91]. X.Y. Liu, H. Yang, J. Gao, and C. D. Wang, "FinRL: Deep reinforcement learning framework to automate trading in quantitative finance," in *Proceedings of the Second ACM International Conference on AI in Finance*, 2021, pp. 1-9.

[92]. H. Buehler, L. Gonon, J. Teichmann, B. Wood, B. Mohan, and J. Kochems, “Deep hedging: Hedging derivatives under generic market frictions using reinforcement learning”, Swiss Finance Institute Research Paper 19-80, 2019.

[93]. X.-Y. Liu, Z. Xiong, S. Zhong, H. Yang, and A. Walid, “Practical deep reinforcement learning approach for stock trading”, NeurIPS Workshop (2018).

[94]. H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading: An ensemble strategy," in *Proceedings of the first ACM International Conference on AI in Finance*, 2020, pp. 1-8.

[95]. N. Vadori, S. Ganesh, P. Reddy, and M. Veloso, "Risk-sensitive reinforcement learning: a martingale approach to reward uncertainty," in *Proceedings of the First ACM International Conference on AI in Finance*, 2020, pp. 1-9.

[96]. Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017.
[97]. S. Bekiros, "Fuzzy adaptive decision-making for boundedly rational traders in speculative stock markets," vol. 202, no. 1, pp. 285-293, 2010.
[98]. Y. Zhang and X. Yang, "Online portfolio selection strategy based on combining experts' advice," vol. 50, pp. 141-159, 2017.
[99]. Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *2017 Intelligent Systems Conference (IntelliSys)*, 2017, pp. 905-913: IEEE.
[100]. Samuel, A. L. 1959. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development 3:211–229. Reprinted in E. A. Feigenbaum and J. Feldman, editors, Computers and Thought, McGraw-Hill, New York 1963.
[101]. J. A. Boyan, *Modular neural networks for learning context-dependent game strategies*. University of Cambridge. Computer Laboratory, 1992.
[102]. G. J. Tesauro, D. Lippman, and S. Hanson, "Practical issues in temporal difference," pp. 259-266, 1992.
[103]. N. N. Schraudolph, P. Dayan, and T. J. Sejnowski, "Using the td (lambda) algorithm to learn an evaluation function for the game of go," vol. 6, 1994.
[104]. M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*: Elsevier, 1994, pp. 157-163.
[105]. D. Silver, R. S. Sutton, and M. Müller, "Reinforcement Learning of Local Shape in the Game of Go," in *IJCAI*, 2007, vol. 7, pp. 1053-1058.
[106]. M. Campbell, A. Hoane, and F. Hsu. Deep Blue. Artificial Intelligence, 134:57–83, 2002.
[107]. J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron, "A World Championship Caliber Checkers Program," vol. 53, no. 2-3, pp. 273-289, 1992.
[108]. B. Sheppard, "World-championship-caliber Scrabble," vol. 134, no. 1-2, pp. 241-275, 2002.
[109]. D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," vol. 529, no. 7587, pp. 484-489, 2016.
[110]. D. Silver *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017.
[111]. D. Wu, Y. Lei, M. He, C. Zhang, L. Ji, and M. Computing, "Deep reinforcement learning-based path control and optimization for unmanned ships," vol. 2022, pp. 1-8, 2022.
[112]. H. Xu, N. Wang, H. Zhao, and Z. J. C.-P. S. Zheng, "Deep reinforcement learning-based path planning of underactuated surface vessels," vol. 5, no. 1, pp. 1-17, 2019.
[113]. D.H. Chun, M.I. Roh, H.-W. Lee, J. Ha, and D. Yu, "Deep reinforcement learning-based collision avoidance for an autonomous ship," vol. 234, p. 109216, 2021.
[114]. K. Duan, S. Fong, and C. L. P. Chen, "Reinforcement learning based model-free optimized trajectory tracking strategy design for an AUV," vol. 469, pp. 289-297, 2022.
[115]. X. Lu, C. Zhai, V. Gopalakrishnan, and B. Buchanan, "Automatic annotation of protein motif function with Gene Ontology terms," vol. 5, pp. 1-11, 2004.
[116]. V. N. Sichkar, "Reinforcement learning algorithms in global path planning for mobile robot," in *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, 2019, pp. 1-5: IEEE.
[117]. B. Wang, Z. Liu, Q. Li, A. J. I. R. Prorok, and A. Letters, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," vol. 5, no. 4, pp. 6932-6939, 2020.
[118]. R. Smierzchalski and Z. Michalewicz, "Path planning in dynamic environments," in Innovations in Robot Mobility and Control: Springer, 2005, pp. 135-153.
[119]. S. Carta, A. Corriga, A. Ferreira, A. S. Podda, and D. R. Recupero, "A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning," vol. 51, pp. 889-905, 2021.

[120]. S. Balhara, N. Gupta, A. Alkhayyat, I. Bharti, R.Q. Malik, S.N. Mahmood and F. Abedi, "A survey on deep reinforcement learning architectures, applications and emerging trend."*IET Communications*, pp: 1-16. 2022. doi: 10.1049/cmu2.12447.

[121]. M.-L. Li, S. Chen and J. Chen, "Adaptive Learning: A New Decentralized Reinforcement Learning Approach for Cooperative Multiagent Systems." *IEEE Access*, vol.8, pp: 99404-99421. 2020. doi: 10.1109/ACCESS.2020.2997899.

[122]. R. Upadhyay, R. Phlypo, R. Saini and M. Liwicki, "Sharing-to-learn and learning to share; Fitting together Meta, Multi-Task, and Transfer Learning: A meta review." *Journal of IEEE Transactions on Artificial Intelligence*, vol. 00, no. 0, pp: 1-21. 2023. doi: 10.48550/arXiv.2111.12146.

[123]. D. Yang, X. Qin, X. Xu, C. Li and G. Wei, "Sample Efficient Reinforcement Learning Method via High Efficient Episodic Memory." *IEEE Access*, vol. 8, pp: 129274-129284, 2020. doi: 10.1109/ACCESS.2020.3009329.

[124]. D.W. Jeong, S.J. Yoo and Y.H. Gu, "Safety AARL: Weight adjustment for reinforcement-learning-based safety dynamic asset allocation strategies." *Expert Systems with Applications*, vol. 227, pp: 1-13. 2023. doi: 10.1016/j.eswa.2023.120297.

[125]. Y. Yu, "Towards Sample Efficient Reinforcement Learning." *Proceeding of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, pp: 5739-5743, 2018.

[126]. S.E. Ada, E. Ugur and H.L. Akin, "Generalization in Transfer Learning." *arXiv preprint*, pp: 1-19. 2019. doi: 10.48550/arXiv.1909.01331.

[127]. M. Xu, Z. Liu, P. Huang, W. Ding, Z. Cen, B. Li and D. Zhao, "Trustworthy Reinforcement Learning Against Intrinsic Vulnerabilities: Robustness, Safety and Generalizability." *arXiv preprint*, pp:1-36. 2022. doi: 10.48550/arXiv.2209.08025.

[128]. P. Ladosz, L. Weng, M. Kim and H. Oh, "Exploration in deep reinforcement learning: A survey." *Information Fusion*, vol. 85, pp: 1-22. 2022. doi: 10.1016/j.inffus.2022.03.003.

[129]. J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu and Z. Wang, "Exploration in Deep Reinforcement Learning: From Single-Agent to Multi-Agent Domain." *IEEE Transactions on Neural Networks and Learning Systems*, pp: 1-24. 2023. doi: 10.1109/TNNLS.2023.3236361.

[130]. A. Plaat, W. Kosters and M. Preuss, "Model-Based Deep Reinforcement Learning for High-Dimensional Problems, a Survey." *arXiv preprint*, pp: 1-22. 2020. doi: 10.48550/arXiv.2008.05598.