IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 January 2023, accepted 25 January 2023, date of publication 2 February 2023, date of current version 9 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3241881

TOPICAL REVIEW

Offloading Mechanisms Based on Reinforcement Learning and Deep Learning Algorithms in the Fog Computing Environment

DEZHEEN H. ABDULAZEEZ[1] **AND SHAVAN K. ASKAR**[2]
[1]Department of Computer Science, University of Duhok, Duhok 42001, Iraq
[2]Erbil Technical Engineering College, Erbil Polytechnic University, Erbil 44001, Iraq
Corresponding author: Dezheen H. Abdulazeez (dezheen.abdulazeez@gmail.com)

ABSTRACT Fog computing has emerged as a computing paradigm for resource-restricted Internet of things (IoT) devices to support time-sensitive and computationally intensive applications. Offloading can be utilized to transfer resource-intensive tasks from resource-limited end devices to a resource-rich fog or cloud layer to reduce end-to-end latency and enhance the performance of the system. However, this advantage is still challenging to achieve in systems with a high request rate because it leads to long queues of tasks in fog nodes and reveals inefficiencies in terms of delays. In this regard, reinforcement learning (RL) is a well-known method for addressing such decision-making issues. However, in large-scale wireless networks, both action and state spaces are complex and extremely extensive. Consequently, reinforcement learning techniques may not be able to identify an efficient strategy within an acceptable time frame. Hence, deep reinforcement learning (DRL) was developed to integrate RL and deep learning (DL) to address this problem. This paper presents a systematic analysis of using RL or DRL algorithms to address offloading-related issues in fog computing. First, the taxonomy of fog computing offloading mechanisms based on RL and DRL algorithms was divided into three major categories: value-based, policy-based, and hybrid-based algorithms. These categories were then compared based on important features, including offloading problem formulation, utilized techniques, performance metrics, evaluation tools, case studies, their strengths and drawbacks, offloading directions, offloading mode, SDN-based architecture, and offloading decisions. Finally, the future research directions and open issues are discussed thoroughly.

INDEX TERMS Fog computing, Internet of Things (IoT), offloading, reinforcement learning, deep reinforcement learning.

I. INTRODUCTION

The exponential growth in the number of IoT devices, estimated to reach approximately 75 billion by 2025, will result in the generation of huge amounts of data (big data) [1]. Big data must be saved, transmitted, and analyzed to be converted into meaningful information that the user can understand. However, most IoT end devices are known for their limited storage and computational capabilities; therefore, storing and processing data on these devices is not an option. Cloud computing technology provides unlimited storage and a large processing capacity via a large number of powerful virtual servers, which can be utilized for applications that are not time-sensitive and do not require a higher level of responsiveness. Nevertheless, because of the centralized fashion, cloud computing technologies are unable to meet the needs of real-time processing applications, owing to the large end-to-end delay and high network bandwidth utilization. Moreover, the centralized approach used by cloud computing leads to increasing delays, which cannot

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

be tolerated by most IoT applications, including virtual reality (VR) [2], augmented reality (AR) [3], autonomous vehicles [4], industrial IoT, and e-health. To satisfy the intensive computational demands and stringent latency requirements of such applications, fog computing has been developed as a potential solution for transferring storage, networking, and processing abilities to the edge of a network [5]. The primary objective of fog computing is to bring cloud-like services to the network edge to optimize the performance of the system in terms of service latency, network bandwidth, resource utilization, and workload balancing [6].

However, deploying fog computing poses additional issues concerning decisions regarding whether tasks should be performed locally or transferred to the fog or cloud. Offloading is essentially the movement of resource-intensive tasks to different platforms to accomplish tasks more efficiently. Therefore, offloading must satisfy different constraints, including latency, load balancing, privacy, and storage. Nevertheless, the distribution of the workload across complicated heterogeneous fog devices, with varying computational resources and capacities, is a major challenge for such fog offloading solutions. The challenge is increased by the growing number of requests, which likely causes the task queues of the powerful fog nodes to become longer. Owing to the increased waiting time of a long queue, the requirements for time-critical applications may be exceeded [5].

In addition, many single fogs are incapable of handling resource-intensive tasks because of their limited processing capabilities, lack of available resources, or dynamic nature of resource needs. Therefore, when a task's processing needs exceed the capabilities of fog platforms, it will be offloaded to the remote cloud server via the vertical offloading mechanism. Another option is a horizontal offloading. This is primarily achieved by sending computing tasks to the best surrogate fog nodes [6]. Making an intelligent decision between horizontal and vertical offloading is essential so that time is not wasted, and attempting to move a load horizontally when vertical offloading is more effective. For example, if a fog node becomes overloaded and unable to handle the load by itself, it can horizontally offload computations to several less loaded fog modes in the same fog layer, rather than vertically offloading to a stronger FN in a higher fog layer or to the cloud. This will assist in lowering the network traffic and latency.

Four major decisions commonly need to be made throughout the offloading process: what tasks should be offloaded (what), where to execute them (locally or remotely) (where), which slot should a task be offloaded (when), and how the task should be offloaded or through which channel (how) [7]. The offloading process can be made more efficient by making high-quality, timely decisions based on what, where, when, and how. For high quality based on the aforementioned decisions, efficient resource scheduling and allocation are required to satisfy the quality of service (QoS) requirements.

The majority of the currently available conventional approaches are considered effective resource management solutions for distributing, scheduling, and executing tasks in dedicated computing situations. These approaches are not useful and efficient in fog environments because of the large number of resources and tasks that are mainly dynamic in nature, such as the Goldman algorithm, Banker's algorithm, Haas-Mohan algorithm, and Isloor-Marsland algorithm [8]. In other words, they do not have a common framework for studying resource management problems in the real-world computing environment, which involves multiple parameters to obtain effective solutions, including mobility, heterogeneity, federation, QoS management, interoperability, and scalability [9]. Different Methods designed to handle this issue are classified as classical optimization, game theory, metaheuristics, and machine learning.

Machine learning, including reinforcement learning (RL) and deep reinforcement learning (DRL) has been widely investigated and studied in the literature to solve offloading issues and other resource management concerns in different uncertain computing contexts. These strategies help make better decisions regarding when, where, what, and how to offload. For example, task offloading approaches based on RL methods have been proposed for fog computing scenarios to determine the optimal decision on where and when to offload a task [10]. Furthermore, an RL-based load balancing algorithm was proposed and deployed in the dynamic fog context to lower the failed allocation probability and reduce the average processing delay [11]. Owing to the large and complex action and state spaces of wide-area wireless networks, traditional reinforcement learning (RL) algorithms may not be capable of determining the optimal strategy within an acceptable amount of time. Furthermore, in large- space situations, such as vehicular networks, the state space is large, and the agent cannot analyze every action in each state to guarantee success. In such situations, the agents are required to generalize the state space. Some states are rarely visited. Reinforcement learning cannot handle continuous or high-dimensional spaces.

As an alternative solution, deep reinforcement learning (DRL) was created by combining reinforcement learning (RL) and deep learning (DL) [12]. DRL algorithms can achieve exceptional performance in complicated control domains, proving that they are better for making decisions in complex and uncertain situations. For example, the authors of [13] solved a task allocation problem based on DRL methods in a dynamic vehicular environment. Another study solved the problem of computation offloading and resource allocation by using a DRL algorithm to reduce system delay [14].

To the best of the authors' knowledge, no comprehensive survey or systematic review paper has been conducted on research articles that used RL and DRL techniques to address offloading concerns in fog scenarios. The goal of this study is to analyze existing research that uses RL and DRL methods to solve offloading problems in fog computing in a systematic and comprehensive manner.

In brief, the following are the major contributions of this review:

- Exploring different survey articles on RL or DRL-based offloading mechanisms in fog and discussing the strengths and drawbacks of each.
- A systematic and comprehensive review of the latest RL and DRL approaches in the field of fog offloading.
- A new taxonomy of current RL and DRL algorithms was developed for offloading mechanisms in fog computing.
- Finally, open issues and future research directions are discuss as well as weakly covered research challenges to enhance offloading mechanisms in the context of fog computing.

The remainder of this paper is organized as follows. Section II reviews the essential related work and motivation. Section III provides the necessary background on fog computing and offloading mechanisms, as well as an overview of the RL and DRL principles. The paper selection and research methodology are presented in Section IV. Section V presents a systematic and comprehensive review of the existing studies that leverage RL and DRL algorithms in the context of fog computing. An analytical discussion is presented in section VI. Section VII discusses corresponding open issues and research challenges for future studies. Finally, conclusions are presented in Section VIII.

II. RELATED WORK AND MOTIVATION

This section presents the current survey and review papers on offloading issues in fog computing. The main advantages and drawbacks of each study were then discussed.

A comprehensive survey of edge computing environments regarding offloading metrics was presented which included quality of service (QoS) and quality of experience (QoE), energy consumption, resource scheduling approaches, performance, gaming theory, and overhead, to make the most suitable decision for computation offloading. This article includes a sufficient number of recently published high-quality studies, and is reasonably organized. However, the article suffers from the following weaknesses: the selection criteria for the included papers were not well defined, the study was not a systematic review, and open issues and future directions were incomplete [7]. Another study provides a comprehensive systematic literature review (SLR) of recent studies focused on resource management approaches in fog computing environments [15]. These studies have been categorized into six classes: resource scheduling, resource allocation, task offloading, application placement, load balancing, and resource provisioning techniques.

In addition, previous studies have provided a comprehensive review of offloading methods in mobile edge computing from the perspectives of game-theoretic offloading methods [16], machine-learning offloading methods [17], and stochastic-based offloading methods [18]. As a strength, these review articles are well structured and follow the standards and format of a systematic study, and they include up-to-date published papers on related studies. However, their surveys did not cover all the reinforcement learning methods. This study is also directly related to mobile edge computing, and the concept of fog computing has not been emphasized. In [19], a comprehensive overview of research areas related to offloading modeling in edge computing was presented by analyzing various types of offloading modeling based on different patterns, such as game theory, (non-)convex optimization, and machine learning. Markov decision process, or Lyapunov optimization. However, the selection criteria for the included papers were not well defined, and the study was not a systematic review. Besides, it does not include future directions.

Furthermore, the authors of [20] provided a survey on computational offloading in edge computing, including different offloading scenarios (IoT devices, between edge servers, and the cloud), influencing factors (device, network, service, and user factors), and offloading strategies (partial and full). During the offloading process, they also discussed key issues, such as whether, what, and where to offload. However, the selection criteria for the included papers were not well defined, and the study was not a systematic review. Besides, it did not include future directions or open issues, which were measured as a core part of the surveys.

Another study provides a survey of recent studies on fog environmenst from the perspectives of argument reality applications [21], smart city applications [22], and healthcare applications [23]. However, these studies did not consider offloading in fog settings. In addition, a taxonomy of optimization techniques and their applications in fog computing environments was presented in [24]. They classified the related research articles into three categories: integer programming, heuristics, and metaheuristics. Examples of applications of optimization techniques found in the reviewed literature include allocation, scheduling, offloading, placement, load balancing, selection, resource provisioning, resource management, migration, clustering, and a combination of these optimization problems.

The authors [25] presented a comprehensive study of offloading mechanisms, focusing on current research papers on offloading strategies in a fog environment, including its architecture, application, and technologies. The selected studies were classified into four main categories: methods based on storage, methods based on computation, methods based on energy, and hybrid methods. Then, it provides a comparison between different offloading mechanism metrics, algorithm types, and evaluation based on the categorization of the selected papers. Moreover, the work in [26] reviewed the literature on fog computing simulation tools and was intended to aid researchers in exploring and evaluating fog-related solutions.

The work in [5] surveyed the existing literature on RL applications for solving resource allocation problems in fog computing environment. In future studies, open issues and challenges should be addressed. However, the selection criteria for the included papers were not well defined, and the study was not a systematic review. Besides, this study focuses on resource allocation rather than offloading issues in fog computing. In addition, [27] provided a detailed

overview of the current applications of RL and DRL to handle various problems in vehicular networks. These schemes are divided into two classes: vehicle management and vehicle infrastructure management. However, this study only considered vehicular networks as a case study in their review paper. Besides, the selection criteria for the included papers were not well defined, and the study was not a systematic review.

Furthermore, in [28], the most recent findings on service placement computation and offloading in fog were presented. Specifically, this study suggests a novel category of optimization techniques for tackling service placement issues in IoT applications running on fog nodes. The methods and optimization objectives play a significant role in this categorization. However, reinforcement learning methods were not investigated as a solution to the offloading issues in fog computing in this study. In addition, the authors of [29] reviewed resource management strategies in fog computing environments considering heterogeneity, resource limitations, and unknown future fog computing traffic. In addition, they presented significant management issues including resource scheduling, resource allocation, task offloading, and resource provisioning. Nevertheless, the selection criteria for the included papers were not well defined, and the study was not a systematic review. Table 1 summarizes previous surveys. Each paper lists its publication year, environment, review type, paper selection method, taxonomy, open issues, and year covered.

The weaknesses of most previous surveys and review papers are as follows.

- Some related review papers did not focus on offloading issues in fog computing.
- Some studies have not focused on RL and DRL approaches in solving offloading related issues.
- Some papers did not present any reasonable classifications.
- Some studies did not address open issues and future directions, which are crucial elements in surveys.
- The process and selection criteria for the papers involved have not been well defined in some studies.
- Some papers did not have an adequate number of articles.
- Some studies did not include newly published articles or state-of-the-art studies, especially those published between 2021 and 2022.

The aforementioned limitations motivated us to conduct a systematic and comprehensive review of offloading strategies based on the RL and DRL methods in fog computing.

III. BACKGROUND INFORMATION

This section presents a brief definition of a fog. First, the layered architecture of fog is described. The offloading process is then explained from different perspectives, including offloading decisions, offloading modes, offloading directions, and offloading metrics. Next, the fog computing evaluation are further discussed. Finally, the fundamental concepts of RL and DRL are also explained.

A. FOG COMPUTING

Fog computing is an extremely virtualized platform that delivers storage, computation, and networking resources between IoT devices and conventional cloud server [30]. Fog computing is considered a supplement to cloud computing and was introduced by Cisco [31] to address the shortcomings of cloud computing such as reducing delays, and minimizing network usage [32], [33]. OpenFog Consortium defined fog computing [34] as; ''a system-level horizontal architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things''.

Fog computing is also defined by the authors of articles [15] as; ''Fog computing is a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralized devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third parties. These tasks can be used to support basic network functions or new services and applications that operate in a sandboxed environment. Users leasing part of their devices to host these services get incentives for doing so''.

Fog computing is composed of a large number of fog nodes and is categorized into two main types: resource-rich devices that include cloudlets and IOx cables whereas routers, wireless access point (WAP) end devices, set-top-units, switches, and stations are resource- hungry devices.

B. FOG COMPUTING ARCHITECTURE

According to previous studies [35], [36], and [37], the hierarchical architecture of fog computing comprises the following three main layers:

Edge layer: This is the layer closest to the user device in the physical environment. It contains different IoT smart devices such as mobile phones, self-driving vehicles, humidity sensors, temperature sensors, and CCTV surveillance systems, and so on. These IoT devices create large amounts of data using sensing physical objects or events. Owing to the limited resources of these IoT devices, the sensed data are transferred to the upper tire (for/cloud) for storage and processing.

Fog layer: This is the middle layer between the cloud and edge devices. The fog layer consists of many fog nodes with limited computing or storage capabilities, including routers, switchers, gateways, base stations, access points, and cloudlets. These fog devices are distributed between the end devices and the cloud, such as shopping centers, cafes, parks, bus terminals, and streets. They can be mobile on a moving carrier or static at a fixed location. End devices are usually connected to fog nodes to obtain capabilities for computing and storing the received sensed data. The fog node stores the data of time-sensitive application for a specific time before transferring it to the cloud. To process the data, the fog computing infrastructure is enabled as a supporting middle layer between edge nodes and the cloud to collect, process, and analyze the data at the edge. The cloud is responsible for storing data in its storage when it

TABLE 1. Summary of related surveys.

Ref.	Publication year	Environment	Review type	Paper selection method	Taxonomy	Open issue and challenges	Covered year
[7]	2019	Edge Computing	Survey	Not- Clear	yes	Not presented	Not mentioned
[15]	2019	Fog Computing	SLRs	Clear	Yes	Presented	2014–-February 2019
[16]	2020	MEC	SLRs	Clear	Yes	Presented	2013-2019
[17]	2020	MEC	SLRs	Clear	Yes	Presented	2013 - 2020
[18]	2020	MEC	SLRs	Clear	Yes	Presented	2016-2020
[19]	2020	Edge Computing	Survey	Not- Clear	Yes	Not presented	Not mentioned
[20]	2020	Edge Computing	Survey	Not- Clear	Yes	Not presented	Not mentioned
[21]	2020	Fog Computing	SLRs	Clear	Yes	Presented	up to October 2019
[22]	2021	Fog Computing	Survey	Not- Clear	Yes	Presented	Not mentioned
[23]	2021	Fog Computing	Survey	Not- Clear	Not- Clear	Not presented	Not mentioned
[38]	2021	Fog Computing	SLRs	Clear	Yes	Presented	Not mentioned
[24]	2021	Fog Computing	SLRs	Clear	Yes	Presented	2016–-November 2020
[25]	2021	Fog computing	SLRs	Clear	Yes	Presented	2016–2020
[26]	2021	Fog Computing	Survey	Not- Clear	Not- Clear	Not presented	2015-2020
[5]	2022	Fog computing	Survey	Not- Clear	Not- Clear	Presented	Not mentioned
[27]	2021	6G vehicular networks	Survey	Not- Clear	Not- Clear	Presented	Not mentioned
[28]	2022	Fog computing	Survey	Not- Clear	Not- Clear	Presented	Not mentioned
[29]	2022	Fog computing	Survey	Not- Clear	Yes	Presented	Not mentioned
proposed	2022	Fog computing	SLRs	Clear	Yes	Presented	2019–mid-2022

is no longer required by the fog node [32]. The fog layer played a crucial role in solving long-latency and real-time analysis problems. Fog nodes are connected to cloud servers to achieve powerful storage and computing capabilities. The proxy server enables communication between the fog and the cloud.

Cloud layer: This layer is located at the top layer. The cloud computing layer is composed of several storage devices and high-power servers, which can provide powerful storage and computing abilities to support permanent extensive computation analysis and the storage of a large amount of data. It also delivers various application services, including smart homes, factories, and transportation. This layer plays a vital role in processing large amounts of data, storing data, and managing the platform services and monitoring systems. Conversely, there is a large distance between this layer and the edge layer, particularly in the IoT layer. The fog computing architecture is shown in figure 1.

In this architecture, each end device or IoT object is linked to one of the fog nodes through various available communication technologies (e.g., 3G, 4G, wireless local area network (WLAN), ZigBee, WiFi, and Bluetooth) or wired connections. Fog nodes can be connected to the cloud or other fog nodes using wireless or wired communication technologies. In addition, all fog nodes were linked to the cloud via an IP core network.

C. EDGE COMPUTING (EC) PARADIGMS

Edge computing paradigms can be divided into three types: mobile edge computing (MEC), fog computing, and cloudlet computing. Many academic researchers use these concepts interchangeably, but there are key differences between them. This section introduces the concepts of cloudlets and mobile edge computing. It then describes the main differences between these terms.

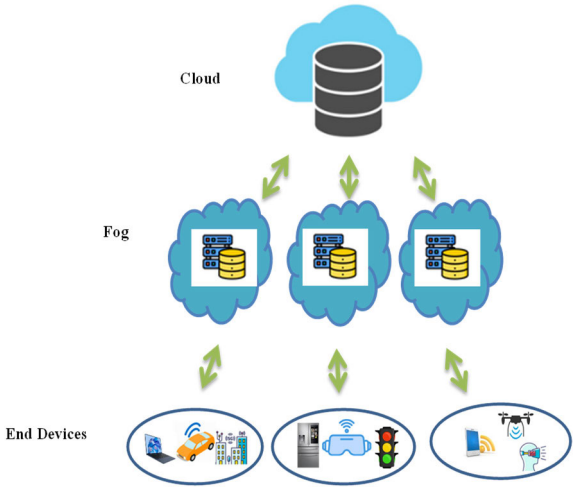

FIGURE 1. Architecture of fog computing.

1) CLOUDLETS

The term ''cloudlet'' was introduced for the first time in 2009 [39]. Cloudlets are small data centers that are normally one hop away from mobile devices and can be accessed via high-speed connectivity such as Wi-Fi, and mobile broadband. The potential benefits of cloudlets include increased bandwidth, decreased latency, offline accessibility, and lower costs [40]. A cloudlet is a reliable, and, powerful computer or a group of computers with a robust internet connection that is used in the proximity of mobile devices. For example, in a classroom or coffee shop.

The architecture of this scheme comprises three layers: mobile devices, cloudlets, and cloud. The purpose of adopting cloudlets is to enable mobile devices with limited storage and computational resources to offload heavy computations to cloudlets, which is particularly useful for time-sensitive applications [41]. Limited coverage is one of the limitations of cloudlets [42]. However, fog computing can overcome this limitation by providing resources to be located anywhere, from end devices to the cloud, and allowing large network sizes.

2) MOBILE EDGE COMPUTING (MEC)

Mobile Edge Computing (MEC) intended to provide cloud computing services at the base stations of cellular networks [43]. The MEC server is a new device that must be deployed close to the base station towers to deliver computing and storage abilities at the edge.

MEC was proposed in 2014 by the European Telecommunication Standards Institute (ETSI) for the first time [44]. In 2017, ETSI officially renamed it MEC for multi-access edge computing. This computing paradigm can be used to provide a wide range of services including augmented reality, location services, video analytics, local content distribution, and caching services. By caching content on the MEC server, it can enable access to local content in real time and with low latency. For example, the authors of [45] proposed reinforcement learning (RL) based energy-efficient MEC collaborative inference scheme to reduce the inference latency and save the MEC energy consumption. According to [24], [35], and [44], the characteristics of the abovementioned edge computing paradigms are summarized in table 2.

Computation offloading in an edge computing environment is different from traditional computation offloading in a mobile cloud environment (MCC). The MCC relied solely on mobile devices and the main cloud server for offloading tasks. Therefore, the main cloud servers may be physically and logically distant from mobile devices, resulting in large response latencies [46]. Edge computing paradigms have been developed to address the latency issue that occurs during the offloading process in MCC. Edge computing is distinguished by the requirement for low latency and the provision of a high workload capacity while being close to user devices. In addition, based on the research article [44] and the features of new edge computing paradigms, computation offloading differs among edge computing technologies.

Regarding proximity to the edge, in a fog computing environment, the fog node may not be the first hop access point for the end device due to the utilization of legacy devices as fog computing nodes (FCNs). For instance, the first router linked to the end device may not have the resources to run an FCN framework. Therefore, the nearest FCN may be several hops away. However, in the case of cloudlets and MEC, the devices communicate directly with the node via Wi-Fi, and at the base station of the mobile network, respectively. They are only a single hop away from the end device.

In addition, fog devices are diverse and heterogeneous in terms of storage, computation, and communication capabilities, such as hubs and, switches. Cloudlet and MEC, on the other hand, do not support heterogeneity devices. Furthermore, the offloading process has changed in fog environments because of the hierarchical and distributed structure of fog computing (IoT, fog, and cloud). On the other hand, MEC and cloudlets have localized structures. The proximity, diversity and heterogeneity of devices and the centralized and distributed architecture of edge computing make the computation offloading process different and more challenging.

D. FOG COMPUTING CHALLENGES

The fog computing paradigm intends to address several IoT and cloud computing application challenges. However, it also has its own challenges that will be discussed in this section.

1) SECURITY AND PRIVACY ISSUES

Because fog computing devices are typically deployed in places that are not under strict surveillance and protection, they may be at risk of traditional attacks that could compromise the system of fog devices in order to carry out malicious tasks such as eavesdropping and data hijacking. Furthermore, privacy and security concerns can be triggered if end users offload computations to neighboring fog servers injected by attackers [47]. Man-in-the-middle, authentication, access control, port scanning, and denial-of-service (DoS) threats

TABLE 2. The characteristics of edge computing paradigms.

Characteristic	Fog	MEC	Cloudlets
Architectural design	Distributed/ Hierarchical	Localized/ Hierarchical	Localized
Storage	Limited	Low	Low
Processing	High	Medium	Medium
Latency	Low	Low	Low
Bandwidth usage	Low	Low	Low
Main computation element	Set-top boxes, routers, switches, ,gateways, access points, base stations, mini servers	MEC server running in base stations	Cloudlets data center in a box
Proximity	One or multiple hops	One hop	One hop
Location for computing	Devices in the path of routing	Base stations and nearby devices	Nearby devices
Access mechanisms	Bluetooth, Wi-Fi, Mobile Networks	Mobile networks	Wi-Fi
Application	IoT application, Big data, Smart home, Smart cities, game video streaming	Mobile applications	Mobile applications
Mobility support	High	Moderate	Moderate
Heterogeneity support	Yes	No	No
Cost	Small (uses legacy or commodity devices)	Large (requiring special devices)	Large (requiring special devices)

have been addressed by a number of proposed solutions for fog computing security challenges [48].

2) RESOURCE MANAGEMENT

Resource management is considered the most challenging issue in fog landscapes because of resource heterogeneity, resource limitations, unpredictability, and the dynamic nature of the fog environment [15]. These issues make it more difficult to manage resource allocation, scheduling, placement, provision, offloading, and sharing. Therefore, effective management solutions are required to satisfy the requirements of these applications.

3) ENERGY MANAGEMENT

Fog computing systems are composed of a large number of geographically distributed nodes and these fog noses has less capability of storage and computing than cloud. Thus, the energy consumption in a fog environment is expected to be greater than that of the cloud. This problem can be reduced by an effective offloading mechanism by sending the energy-intensive parts from fog nodes to other powerful neighboring fog nodes or clouds servers. For instance, the authors of [49] proposed an energy-efficient offloading decision method to simultaneously reduce energy consumption and meet response time constraints. A schedule-delay aware offloading technique was designed in their paper to ensure service quality in real-time jobs and reduce the energy consumption of fog-cloud devices, thereby maximizing device lifetime. Substantial research is required to develop successful energy management technologies in fog environment [50].

4) QUALITY OF SERVICE (QoS)

Fog systems consider various QoS metrics when designing a successful system. According to [51], 11 types of QoS metrics were defined (response time, deadline, throughput, resource utilization, execution time cost, energy consumption, availability, reliability, security, and scalability). Depending on the application requirements, these metrics were considered. Trade-offs between different QoS metrics are often necessary when implementing QoS provisioning, which is made more challenging owing to the dynamic nature of different applications and their requirements.

E. OFFLOADING

Offloading is an efficient mechanism that migrates computation or data from a resource-constrained device to another powerful device or fog/cloud to improve the system performance, particularly for computation-intensive or latency-sensitive applications. This mechanism offers various benefits such as reduced latency, decreased device energy consumption, and improved QoS parameters. Various aspects must be considered when making offloading decisions, including performance maximization, delay minimization, and energy consumption minimization.

F. OFFLOADING DECISIONS

Before offloading the computation, several decisions must be made [52], such as

1) WHAT TO OFFLOAD?

How much of the workload should be offloaded? Or what to offload? The task scheduler is responsible for deciding whether the task can be offloaded, and if so, in what capacity (i.e., partial or full offload).

2) WHEN IS OFFLOADING?

The task scheduler is responsible for selecting the appropriate time period for offloading the tasks. The offload interval is chosen by the task scheduler under different constraints.

3) WHERE IS TO OFFLOAD?

This question is related to the location of execution (either locally or remotely). If remote, the optimal fog node for offloading should be selected among the available nodes.

4) HOW TO OFFLOAD?

Through which channel or through which path to offload.

5) WHAT KIND OF OFFLOADING STRATEGY WILL BE USED?

What is the main goal of offloading, maximizing or minimizing a single performance metric, joint optimization, and the trade-off among multiple objective metrics?

Different techniques have been suggested to make better decisions with respect to computational tasks and data (what) to offload, place to execute (either locally or remotely) (where), what point in time to offload (when), and through what channel (how) to offload. High-quality and timely offloading decisions are based on what, weather, where, when, and how questions help to enhance the efficiency of the offloading process. To achieve high-quality based on the above mentioned decisions, effective resources management solutions are required to satisfy the requirements of the applications.

A crucial aspect of computation offloading is deciding where to offload the task (e,g., fog, cloud, or a combination of these). Here, task scheduler is required to determine which tasks will be handled by the edge, fog, and cloud layers in order to meet the desired design goals. Generally, computation offloading methods handle computing resource restrictions such as the storage space of IoT devices, power, battery backup, and sensors. Resource allocation refers to the algorithm used to allocate computing tasks among different fog nodes under different QoS requirements and other restrictions. Several strategies have been introduced, which can be categorized into auction-based and optimization-based approaches. The proposed auction-based solutions are based on market pricing mechanisms that consider the demand and supply of fog nodes. Resource allocation approaches based on optimization, on the other hand, match cloud servers and fog nodes for IoT users [29]. For instance, [53] proposed a resource allocation method based on priced timed petri nets in a fog computing setting.

Task allocation and scheduling are considered the most challenging and important step in the computation offloading process of the fog computing environment because of resource limitations, unpredictability, resource heterogeneity, and the dynamic nature of the fog environment [15]. Methods designed to handle this issue are classified as classical optimization, game theory, metaheuristics, and machine learning. Machine learning, including the RL and DRL algorithms, is widely considered an efficient alternative to traditional optimization strategies. These methods make more intelligent decisions while offloading tasks to cloud or fog servers. In addition, resource allocation concerns such as CPU cycles, channel access, and time allocation, can be addressed using these approaches. The effectiveness of computation offloading is improved by identifying suitable nodes and appropriate resource allocation strategies that satisfy the QoS requirements of applications.

G. OFFLOADING MODE

Typically, mobile apps can be split into a series of coarse-grained or fine-grained tasks, each of which comprises sequential and parallel components. In fine-grained offloading, only a small proportion of the task is sent to a higher-powered computational device, whereas in coarse-grained offloading, the entire job is sent [54]. It may not always be advantageous to offload all the computing components to a remote cloud. During the offloading process, the device may waste more energy and time than locally. In addition, the cost of the transfer may not be insignificant if an unsuitable part is chosen for offloading.

Therefore, we must intelligently select which parts of the application should be executed on a remote cloud server and which parts should remain on the mobile device to decrease response time and energy consumption. However, some applications are relatively basic or strongly integrated and cannot be separated into multiple tasks for parallel processing [20]. They must be completely offloaded to a server or run on a local device. The fog contains two computational offloading modes: full (or binary) and partial offloading. In binary offloading, the tasks of the application cannot be partitioned and must be executed as a whole, either locally or offloaded to the server [20], [55]. With partial offloading, application tasks can be subdivided into several components. These components are then executed locally or offloaded to the servers [55]. The application can be partitioned statically or dynamically as follows:

1) STATIC PARTITIONING

It is decided in advance which application components should be executed locally and which should be offloaded [20]. For instance, software programmers apply static annotations (e.g., @offloadable or @Remote) to methods that indicate that they should be offloaded. However, programs always have non-offloadable components that must be processed locally, such as face detection, user input, and positioning.

2) DYNAMIC PARTITIONING

A task's resource needs may change based on its input data and user-defined objectives (e.g., battery consumption and, response time). In addition, resource availability may change in wireless network (e.g., network latency and, bandwidth) and service nodes (e.g., memory and, available CPU power). To adapt to various network conditions, latency limitations, and server states, appropriate partitioning decisions must be dynamically determined during the runtime [52].

H. OFFLOADING METRICS

In this section, the most important metrics used in the computational offloading domain are briefly explained. These offloading metrics are selected in accordance with the requirements of the application. Trade-offs between different metrics are often necessary when implementing successful applications.

Energy consumption: The offloading mechanism is consumed the total energy. This mechanism sends the task

from the end device to the server, runs a task on the server, and returns the results to the end device [56]. In this context, various considerations play an essential role in consuming energy in fog computing including the data size to be offloaded, the wireless channel's transmission rate, power, gain, and bandwidth, as well as the CPU's power consumption per cycle and the required CPU cycle of the IoT device per task bit.

Latency: The total time taken from transferring the task to the edge servers, running on the server, and receiving the response from the server is referred to as task execution latency [57]. Some parameters have a significant impact on the total delay, such as the data size of the task to be offloaded, wireless channel bandwidth, CPU rate of the edge servers, and the number of CPU cycles required to process each byte of incoming data.

Cost: In the offloading scope, the total execution cost comprises local and remote execution costs, considering both processing and buffering delays. This cost metric depends on the task demand, response time of the task, and location of the task deployments [58].

Response time: Response time is presented as the total time required to offload a task from the local device to a remote server and to receive a suitable answer on the local device. The response time differs from the latency of the system. Latency is the amount of time taken for a request to be sent and then received at its destination, whereas response time is the amount of time between sending a request and receiving a suitable answer [59].

I. OFFLOADING DIRECTION

This section investigates where the computational tasks may be offloaded. Offloading computational tasks can occur in a variety of locations, including IoT devices, fog servers, and clouds. The offloading destination is determined by the trade-off between influencing factors and several objectives. Therefore, the offloading location is crucial because it determines the algorithms to be performed [60]. Such offloading is required when a service provider's assigned task exceeds its processing capacity, and must be transferred to another service provider with adequate computing power [6]. Offloading can be done in three different ways [6]: horizontally, vertically, or in a hybrid topology.

1) HORIZONTAL OFFLOADING

Horizontal offloading occurs when two entities belong to the same tier, such as a cloud-cloud edge-edge, fog-fog, or cloud-cloud.

2) VERTICAL OFFLOADING

Vertical offloading always occurs between two entities belonging to different tiers, such as edge-fog, edge-cloud, edge-fog or fog-cloud.

3) HYBRID OFFLOADING

Hybrid offloading is a mix of horizontal and vertical offloading in which entities can offload horizontally with another entity in the same tier, while also offloading vertically with another entity in another tier. For instance, in a fog-fog-cloud, fog is offloaded with another fog in tier-2 and also offloaded with a cloud in tier-3.

J. EVALUATION OF FOG COMPUTING SOLUTIONS

The fog computing testbed has the potential to enhance the evolution of fog computing. However, it is challenging to conduct large-scale real-world experiments in a fog computing context because of implementation time and high cost. Therefore, few studies have evaluated the experimental results in realistic and large scale areas. For instance, in [61], a multilayer (IoT-device, edge, fog, and cloud) streaming analytics platform with a two-tier fog layer consisting of analytics and streaming tiers was proposed. This approach, which has been functionally verified on a testbed consisting of clusters of low-cost off-the-shelf components, illustrates the feasibility of both large-scale data analytics and real-time streaming processing. The objective is to manage a large number of cyber physical systems applications based on streaming and big data analytics IoT.

Another study [62] proposed a time-changing graph to evaluate the competency of vehicular fog computing (VFC) by analyzing Beijing's vehicular mobility trace and constructing a mode under diverse scenarios in a large-scale urban mobility environment. Furthermore, in [63], a case study on the creation and deployment of applications in a large-scale, dynamic fog computing setting utilizing an open-source platform distributed node-RED (DNR) was presented. Solutions for dynamic and scalability-nature prototypes were created using the Omnet++ network simulator. In addition, the work [64] a self-similarity-based load balancing (SSLB) approach for large-scale systems in a fog computing setting was presented. To enhance SSLB efficiency, they presented an adaptive threshold strategy and a corresponding scheduling algorithm.

Because fog nodes are distributed, it is difficult and expensive for researchers to develop a testbed prototype. Some of them can manage it, but it is expensive to build a large fog/edge computing infrastructure. Therefore, it is important to create a fog/edge testbed as a service. An appropriate real network testbed is necessary to assist researchers in testing their experiments, ideas, and algorithms in realistic fog computing environments. It can also benefit researchers by decreasing the deployment time and saving costs. The piFogBed was introduced in [65] as the first fog computing testbed based on a real network and devices, which enables users to execute real fog applications and receive more realistic results by utilizing Docker container and Raspberry Pi technologies.

Furthermore, the study in [66] introduced Fogbed, an open source fog simulation system based on Docker and Mininet. The existing components were developed for rapid prototyping and testing of fog services in a real desktop environment. Fogbed enables developers to use their verified applications in real-world settings, with minimal modifications. FogTestBed was also presented as a framework for

FogTestBed as a service [67]. It enables testbed owners to construct their testbed infrastructure at the network's edge and to provide secure and private access to clients executing experiments or tests."

However, no specialized fog computing testbed exists to date to assist researchers in testing their concepts, designs, prototypes, and distributed algorithms in large-scale realistic fog computing situations. Therefore, researchers typically use existing simulators to validate the efficiency of fog computing offloading solutions under specific use cases and scenarios. For example, iFogSim [68] was the first simulator for fog computing. It is an extension of the most well-known cloud computing simulator, CloudSim [69]. iFogSim allows for the evaluation of resource management techniques based on energy consumption, latency, network usage, throughput, and operational costs. iFogSim provides a graphical user interface for describing fog network topologies, including sensors, actuators, cloud servers, and connections between them.

EmuFog [70] is another piece of edge computing simulation software that allows users to emulate large-scale edge computing networks on individual computers. EmuFog is based on container technology, which means that each device in the emulation network utilizes its own namespace to save execution and network information and run the same program independently. However, some nodes in realistic edge computing situations are dynamic and resource-constrained, and EmuFog is unable to model such situations, making it difficult to obtain accurate experimental results. Furthermore, EdgeCLoudSim [71] is a CloudSim-based edge computing simulator that can assess edge computing performance. Its edge computing situations enable real load generation and mobility with respect to computing load, package size, and the ability of users to combine their own requirements into a tool. However, the outcomes of the experiment differd from those of the real scenario.

The MobFogSim simulator was recently been developed to simulate application migration and device mobility in a fog computing scenario [72]. This is an extension of iFogSim that incorporates mobility features into the fundamental functionality of several iFogSim components. However, the MobFogSim mobility support system only interacts with cloud datacenters and IoT gateways, which limits its ability to create clusters in fog computing settings. In addition, [73] proposed the iFogSim2 simulator, an extension of the iFogsim simulator, to handle service migration for various IoT device mobility models, microservice orchestration, and node clustering in edge/fog computing environments. The benefits of employing simulation-based techniques include the reproducibility of the studies and a less expensive method for evaluating the efficiency of a target platform because physical hardware is not required.

K. REINFORCEMENT LEARNING

Reinforcement learning is defined as a feedback-based machine learning technique in which an agent learns to interact with the environment by executing actions and observing their outcomes. Markov Decision Processes (MDPs) are mathematical models [74] that describe how an agent interacts with its environment. When an RL problem meets the Markov property, the future depends only on the current state and action and not on what happened in history [75]. According to work [76], an MDP is formally defined as a tuple of five components (S,A,P,R, and Y), where S denotes the collection of states, A denotes the collection of actions, P: $S \times A \times S \rightarrow [0, 1]$ denotes the probability of moving from one state to another given a specific action, R denotes the reward function, and Y represents a discount factor that determines the significance of future rewards $Y \in [0, 1]$.

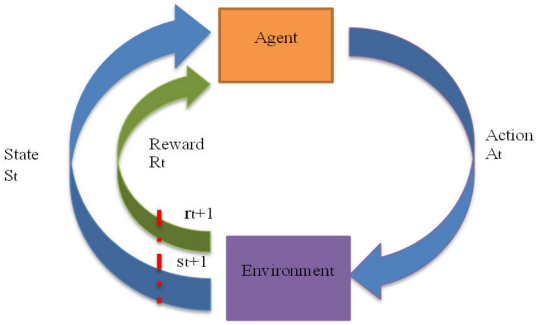

FIGURE 2. Reinforcement learning process.

In order to solve the MDP problem, At the beginning, the agent is in a specific state 's' of the environment, it executes an action 'a'. Once the agent has taken action, the agent moves into the next state 's-' of the environment and receives some reward 'r'. As shown in figure 2, the agent performs this cycle several times for learning.

The agent takes different actions to discover new information and learns by receiving a reward for each correct action and a penalty for each wrong action. Action space refers to the set of options that an agent can execute in a given state. There are two categories of action spaces: discrete and continuous. A discrete action space is used to describe systems where the action space is small, finite, and countable, whereas a continuous action space is used to describe in systems which an action can take any value within a given range [77].

The agent received a reward immediately after performing the action. The immediate reward R is defined by a numeric value (positive or negative) to assess the desirability of the action taken by the agent. Thus, the agent's objective is to optimize cumulative rewards rather than immediate rewards.

In large space problems, such as vehicular networks, the state space is vast, and it is impractical for the agent to continually explore every potential action in each state to achieve the appropriate confidence. An agent must generalize the state space in such situations. Some states may be infrequently visited or may have characteristics that allow for generalization. Such continuous spaces and high dimensions cannot be handled using conventional reinforcement learning. Deep reinforcement learning was used to solve this problem.

Deep reinforcement learning integrates deep learning (DL) and reinforcement learning (RL) [78]. DRL is an enhanced

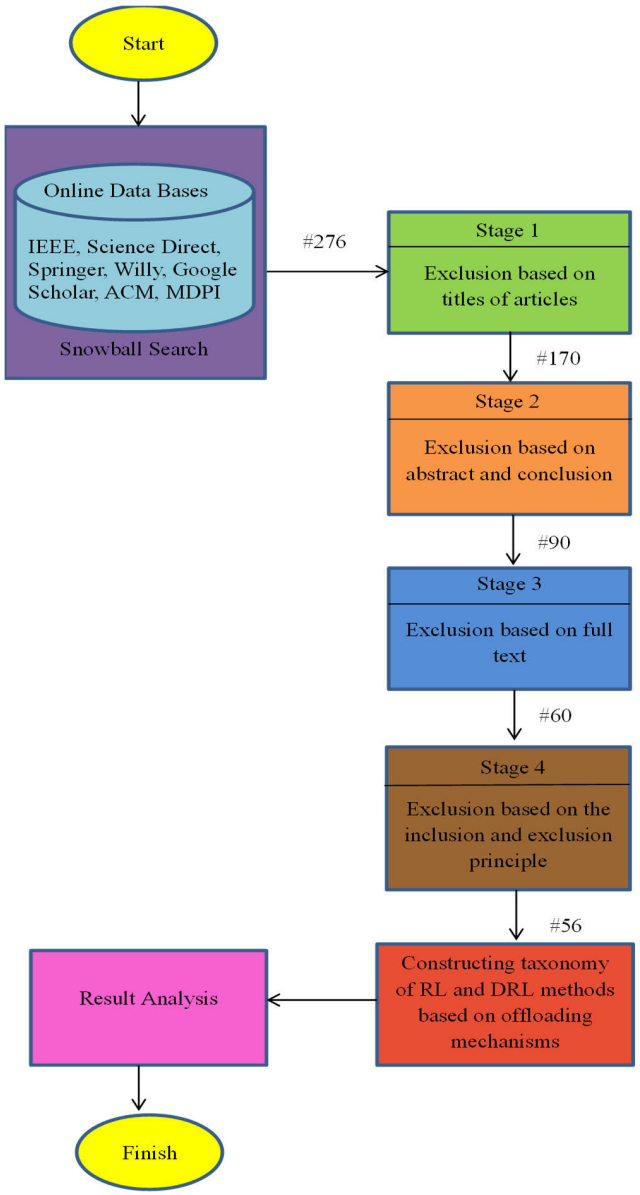

FIGURE 3. Review methodology.

version of the RL technique proposed by DeepMind [79], in which DL is utilized as a powerful tool for enhancing the learning rate of RL techniques [80]. In traditional RL, the values of states and actions are recorded using a tabular method. Therefore, the deep learning architecture in deep reinforcement learning can be considered as a functional approximator, helping the system handle high-dimensional data and provide an approximation in the face of massive actions and state spaces. The mapping from state to action in this learning is defined by neural networks rather than an SVM, decision trees, or other function approximators [45]. Deep learning convolution and recurrent neural networks have also been used [81]. For example, the Q-values for a finite number of discrete actions and states can be represented in a table. However, to represent Q-values in continuous state spaces, function approximators, such as DNNs, are required.

IV. RESEARCH METHODOLOGY

This section presents the methodology used to conduct the comprehensive literature review. First, it formalizes the research question. Next, the electronic database sources used for finding and retrieving papers related to offloading mechanisms based on the RL and DRL methods in fog computing are discussed. Furthermore, it outlines the general and exact keywords used to identify relevant articles and their quality evaluation process.

A. QUESTION FORMALIZATION

The research objective is to search, collect, and identify the best articles on offloading techniques based on RL or DRL approaches in the fog computing field. The following are some of the research questions that will be answered in this study:

RQ1: What categorization can be applied to RL or DRL based offloading mechanisms in fog computing?

RQ2: Which types of algorithms are utilized by RL or DRL based offloading mechanisms in fog systems?

RQ3: Which typical performance metrics are utilized in RL or DRL-based offloading techniques in fog computing?

RQ4: What cases are studies considered in RL or DRL based offloading techniques in the fog paradigm?

RQ5: Which tools are used to evaluate RL or DRL based mechanisms for offloading in fog systems?

RQ6: Which offloading modes are applied in RL or DRL-based offloading techniques in the fog paradigm?

RQ7: What offloading direction is usually applied in RL or DRL-based offloading techniques in fog areas?

RQ8: Which offloading decisions are made with respect to what, where, when, and how are decisions in their strategy based on RL or DRL-based offloading techniques in the fog area?

RQ9: Is SDN incorporated into their strategy based on RL or DRL-based offloading mechanisms in fog computing?

RQ10: How many studies have formulated their offloading strategy problem as an MDP or partially observable Markov decision processes (POMDP) problem and solved it based on RL or DRL approaches in a fog environment?

RQ11: What are the future research directions and open issues for RL and DRL based offloading mechanisms in the fog paradigm?

In Sections VI and VII, the answers to the research questions are discussed.

B. SOURCE OF INFORMATION

The following electronic databases were searched for studies relevant to offloading mechanism-based RL or DRL approaches in fog systems:

- Google Scholar (<www.scholar.google.co.in>)
- IEEE eXplore (<www.ieeexplore.ieee.org>)
- Springer (<link.springer.com>)
- ScienceDirect (<www.sciencedirect.com>)
- WileyInterscience (<www.Interscience.wiley.com>)
- MDPI (<https://www.mdpi.com>)
- ACM Digital Library (<www.acm.org/dl>)

TABLE 3. Automatic search paper filtering using inclusion/exclusion criteria.

Inclusion	Exclusion
1. The publications present assessments, solutions, or experiences for offloading strategy in a fog environment. 2. Studies emphasizing offloading issues in the fog environment. 3. Papers published between 2019 and mid-2022.	1. Study not applying reinforcement learning methods in offloading mechanism. 2. Study related to reinforcement learning methods based on Multi-Armed Bandit (MAB) methods. 3. Study related to edge, cloudlet, MCC or MEC. 4. Non-English research papers 5. Books and book chapters. 6. Review and survey papers. 7. Duplicate paper.

C. PAPER SELECTION PROCESS

This comprehensive literature review has searched the aforementioned digital libraries using relevant keywords such as "fog," "fog computing," "offloading," "reinforcement learning," "Deep learning learning," "DQN," "DDQN," "A2C," "SAC," and "Q learning"., Snowball searches were also performed using the publications that cited each found paper to find further related papers. In addition, this systematic review included research publications written in English between 2019 and mid-2022. A total of 276 research articles from various conferences and journals were identified through the search procedure. As shown in figure 3, the final paper selection was filtered using both inclusion and exclusion strategies. The total number of research publications decreased in four stages: first, based on their titles, to 170; and second, based on their abstracts and conclusions, to 90. In the third stage, 60 research papers were selected based on full-text analysis. Finally, based on the inclusion and exclusion principle shown in table 3, 56 papers were selected to answer the current study technical questions. No research articles found in the MDPI or ACM digital libraries met the inclusion and exclusion criteria.

V. OFFLOADING MECHANISMS IN FOG COMPUTING BASED ON RL AND DRL TECHNIQUES

This section provides a review of current publications on RL and DRL algorithms used to address offloading problems in fog computing. Figure 4 shows the taxonomy in which RL and DRL algorithms are organized into three major classes: value-based, policy-based, and hybrid-based algorithms. Q learning, DQN, Double DQN, Dueling DQN, DRQN, and SARSA are "value-based" approaches, whereas PG, DDPG, and PPO are "policy-based" approaches. Hybrid approaches include DDPG and SAC. This study focuses on model-free approaches to address offloading issues in a fog environment. A summary of the results is presented in table 4.

A. VALUE-BASED ALGORITHMS

The value function, also referred to as the value of the policy, is used to assess the states depending on the cumulative reward the agent obtains over time. Value-based RL approaches use temporal difference (TD) learning to approximate the value function rather than learning the policy explicitly [78]. For each learned policy (π), there are two related value functions: the state-value function 'V(s) and the state-action value function 'Q(s, a). The objective of the value-based method is to identify the optimum value function, which is the maximum value in any given state that can be obtained under any policy. Typical value-based RL algorithms include Q-learning and SARSA.

1) Q-LEARNING

Q-learning is a model-free reinforcement learning technique that can be used to determine the optimum policy for a given state. The Q-learning algorithm learns each state to determine the optimal course of action based on the Q- value function. Learning occurs through trial and error until a particular course of action yields an ideal policy [82].

Several offloading problems in fog computing have been solved with the help of the Q learning method. For example, the authors of the paper [10] proposed an innovative approach to the issue by giving IoT nodes the option to choose to offload tasks to the proximity of fog nodes or the ideal fog node and the remote cloud to meet the needs of the applications. A Q-learning-based algorithm is employed to solve the model and select the optimal offloading policy. The fundamental aim of this study is to reduce the total latency and distribute tasks evenly across fog nodes by lowering the number of offloading operations required to allocate a task to an appropriate fog node. Numerical simulations demonstrate that the recommended technique outperforms other approaches in terms of decreasing latency, processing more tasks, and distributing the workload more smoothly. Furthermore, the authors of work [11] address the load balancing problem while achieving the lowest possible latency in fog networks. To address this issue, a decision-making approach based on Q learning was developed to determine the best offloading action to take while the reward and transition functions are unknown. To reduce the overload probability and processing time, the proposed methods enable fog nodes to choose an available neighboring fog node based on their resource capabilities and to offload the maximum number of incoming tasks. Similarly, in [83], a Q-learning-based technique for selecting ideal offloading nodes in opportunistic edge computing was proposed.

Additionally, a new approach to fog-based IoV network communication security, QoS improvement, and end-to-end delay reduction has been proposed by a study [84]. This system exploits the advantages of SDN and the block-chain technology. The SDN controller uses a Q learning based RL algorithm to assign tasks to fog nodes so that traffic can be distributed more evenly. As a result of the authors' experiments, SaFIoV is capable of avoiding network congestion, reducing latency, and efficiently utilizing network resources. Similarly, the authors in [85] proposed a secure computation offloading scheme to minimize delay and energy consumption in the IoT-Fog-Cloud environment.

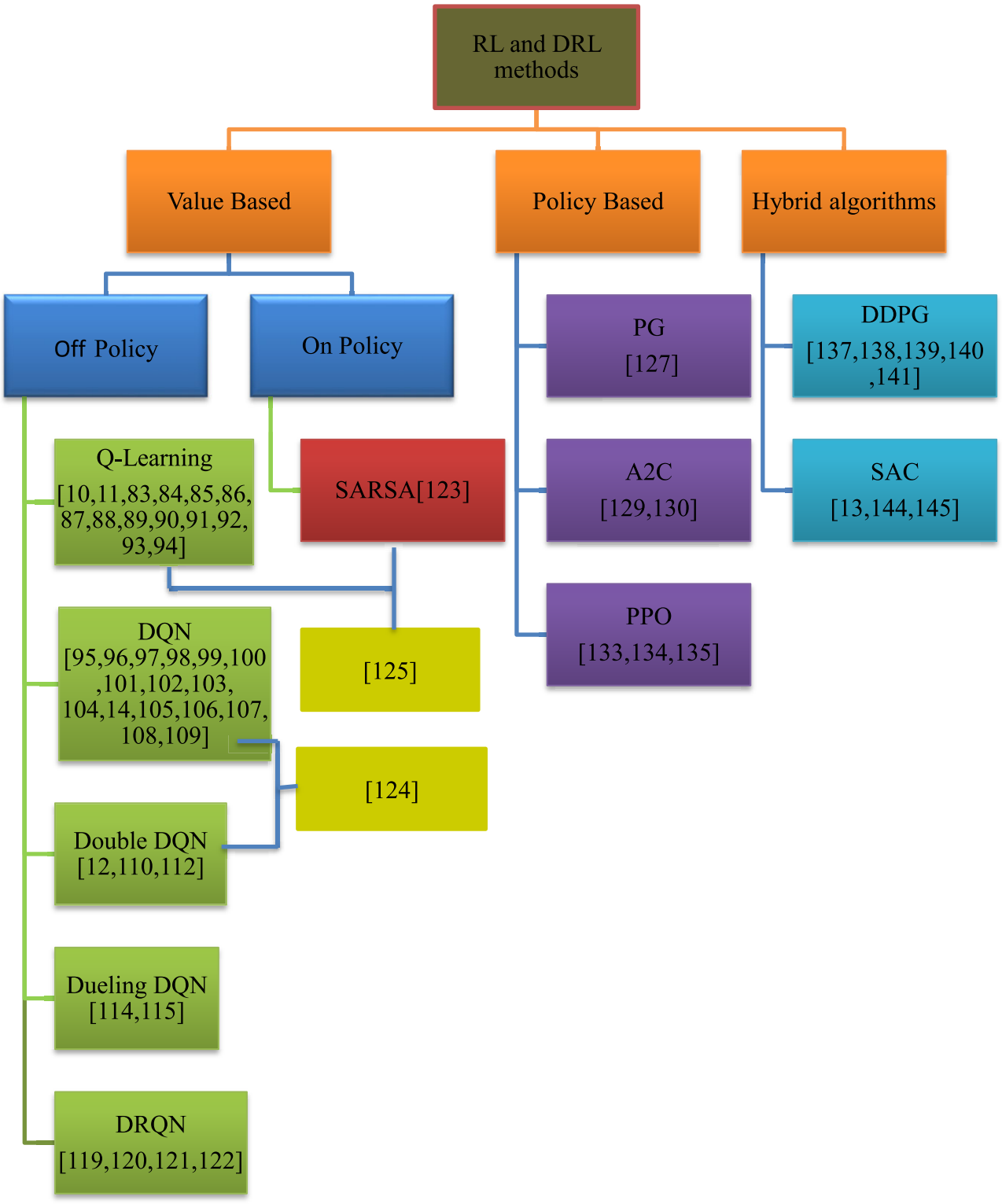

FIGURE 4. Taxonomy of fog computing offloading mechanisms based on RL and DRL methods.

Selection of the best fog node is based on Particle Swarm Optimization (PSO). When the fog node cannot handle the workload within the latency constraints, it is forwarded to the remote cloud server for further processing. The authors propose a Q-learning-based dynamic offloading strategy for this purpose. In the next step, the researchers used a neuro-fuzzy model to secure information at the fog node's smart gateway.

One of the most challenging aspects of resource management for intelligent mobile apps is deciding when and how to offload and provides edge servers to control their dynamic workloads of mobile applications. The authors of [86] used the learning automata technique to make the most optimal offloading selection for workloads submitted by smart mobile applications. In addition, the Q-learning technique and LSTM prediction model were utilized to make an appropriate scaling decision for adding or removing the edge server to adapt to workload fluctuations. Compared with other methods, the proposed technique enhances CPU usage and reduces the energy consumption and execution time. In [87], proposed an innovative framework named ''DroneCOCoNet '' for drone video analytics that enables sophisticated processing of large video data sources utilizing fog computation offloading and accomplishes network protocol selection related to resource awareness. Heuristic and reinforcement learning based Q learning approaches were used to deal with computation offloading problems to reduce overall computation costs and latency in edge/fog resources while also reducing video processing times to fulfill application requirements.

The study in [88] proposed a reinforcement learning-based user access and computation offloading technique in an F-RAN scenario, which used Q learning to maximize system resource utilization and minimize energy consumption by considering both the downlink and uplink task requirements. Furthermore, [89] proposed an intelligent local offloading to enhance the latency and energy efficiency of smart factories. Besides, [90] designed cost-efficient computation offloading (CeCO) for green industrial fog environments. In their research the fog controller concept was introduced to control computation offloading between industrial devices. Initially, industrial devices distinguished between remote and IIoT executable tasks. They also included a frequency-enabled power management strategy to minimize energy emissions in an industrial context. Overburdened tasks were then transferred to the fog controller, where the primary fog controller used a probabilistic technique to select the best fog device. Additionally, the fog controller uses the Q-learning method to determine the shortest path to the destination.

Moreover, [91] proposed a novel framework called (ARTNet) for efficient task offloading in software defined vehicular (SDV F) based on the Q learning method. The major objective of the proposed framework is to improve the offloading action through each system to optimize the utility while optimally distributing the IoV workloads and reducing the processing time. In addition, the authors of [92] presented a technique called GASDEO for offloading a decision-based MAPE to different mobile devices, fog, or cloud architectures. According to the proposed technique, fog devices (FDs) were analyzed locally using a greedy technique, starting with the sibling nodes and continuing to the parent nodes. In the following stage, the DRL method is used to select the best destination for running modules on a mobile device, fog, or cloud environment, in terms of power consumption, total execution cost, and network resource usage. The results indicated that the suggested technique led to improvements over the local execution, First-Fit (FF), and ASDEO methods.

Furthermore, offloading tasks from heterogeneous nodes at the edge of an open and dynamic network is challenging because of the fluctuating workload generated by applications with different service level agreement (SLA) and quality of service (QoS) requirements. The study presented in [93] provided a new learning-based task offloading approach to orchestrate the workload at the network's edge. In comparison with the baseline algorithms, the results indicate that resources are better utilized and tasks are performed with a higher success rate.

2) DEEP Q NETWORK (DQN)

In the context of fog computing, it is challenging to fully understand the system dynamics. Q-learning is a model-free approach for solving this issue. However, Q-learning is a tabular approach, and its size increases exponentially with the number of nodes. Therefore, defining and updating a Q-table in a large state space environment is a complex and challenging task. The deep Q network (DQN) method was used to resolve this problem. In this configuration, rather than generating a Q-table, the neural network estimates the Q-values for each action and state. Many academic researchers have proposed offloading problems in fog computing under MDP, which can be addressed by the DQN, which decreases the number of Q-states that must be trained for each action.

For example, in [94], a computation offloading and resource allocation system for F-RANs was proposed using Deep Reinforcement Learning (DRL). This offloading problem can be described as an MDP and solved efficiently using a DQN. The main idea of the proposed approach is that the DRL controller can make intelligent decisions to process a produced task locally or to transfer the task to a fog access point (FAP) or remote cloud server. The simulation results indicated that the recommended design considerably minimized delays and maximized throughput in the system. Following a similar idea, for long-term reduction in system energy consumption, the authors of [95] proposed a deep reinforcement learning (DRL) approach based on a DQN with a scenario of multiple IIoT devices and multiple fog access points (F-APs). Subsequently, [96] presented a new incentive system for the IoV scenario based on contracts that integrate both resource utilization and resource contribution. The proposed approach was used to determine the best location for executing the modules (mobile devices, fog or cloud). The authors proposed a distributed deep reinforcement learning (DRL) based DQN method to assign resources and reduce the system complexity.

In addition, a deep learning based reinforcement network was presented for the resource allocation and task offloading problem in [97]. Specifically, a deep Q learning network (DQN) was used to address issues modeled as an MDP. The DQN receives the tasks in the queue as the input and the resources available on the fog node. Therefore, instead of using a standard convolutional neural network, this study used a short term memory network (LSTM) in the DQN.

A DQN environment was constructed with fog nodes using the reinforcement learning toolkit in MATLAB and the SUMO environment for VANET behavior. The DQN selects the fog node to be offloaded, and the vehicle must be inside the new node's range of transmission when the simulation is run. Next, the vehicle locations were predicted using the Kalman filter prediction algorithm in the DQN setting.

Moreover, the authors of [98], studied a task offloading scheme in optimal query and policy dynamic environments. The authors were able to acquire the ideal query policy at the task node using the reinforcement learning framework, resulting in long-term optimized task offloading performance. Their research also proposed a simple and, effective approach for the task node to learn the best query decision by utilizing deep Q-networks, whereas the task node is unaware of the dynamics of the system. According to the numerical results, the performance of the proposed query policy learning method was close to the optimum performance. Similarly, in [99] task offloading based deep Q learning in the healthcare Internet of Things (IoT) was used to distribute the load across the edge, fog, and cloud. Besides, the task offloading problem in collaborative vehicle networks was modeled as an MDP by the authors of [100].Then, to enhance user vehicle performance while still meeting URLLC constraints, a Deep Reinforcement learning–based URLLC–aware DQN–based task offloading algorithm DREAM, was developed. The proposed schema outperformed other schemes with existing task offloading methods in terms of queuing delay, throughput, and URLLC. Finally, [101] proposed an offloading strategy based on deep Q-learning to maximize the QoE of tasks in delay-constrained VFC. Considering both the deadline and delay of a task in the reward function of Deep Q-Learning

With the continued growth of extensive IIoT applications, effective service delivery and energy reduction face significant obstacles. However, a single fog device cannot fully perform large-scale applications owing to a lack of resource availability. A partial service provisioning approach offers a potentially successful solution to activate services on several fog devices or to collaborate with cloud servers. Inspired by this scenario, the work [102] presented a cooperative partial service provisioning technique for tackling massive industrial applications in fog networks. First, this study aims to jointly enhance energy consumption and processing latency across all IIoT applications in industrial fog networks. To accomplish this goal, a task partitioning technique for splitting large IIoT tasks into several independent tasks was presented. Then, to intelligently distribute the partitioned tasks across the proximate fog devices, a DRL based DQN enabled service provisioning technique is deployed.

According to [103], task offloading can be solved by considering both communication and computation resources in a mobile vehicle network. The authors formulated a nonlinear issue for energy efficiency. A deep reinforcement learning (DRL)-based DQN approach is used to solve the formulated problem. The authors of [104] proposed a deep Q-learning network (DQN) method to minimize both delay and energy consumption in vehicle fog computing. Offloading decisions in IoV are challenging because of the dynamic nature of the environment and large number of existing state spaces. In addition, [14] proposed a low-complexity deep reinforcement learning technique based on DQN to address the issue of job offloading and resource allocation in a cell-free radio access network. An optimization aim of decreasing the system latency is employed, and a DNN is utilized to speed up the learning of the system, which can make optimal selections under a high number of users and tasks and generate appropriate offloading techniques for computation. Consequently, computational resource allocation and task offloading have been jointly enhanced to address and solve the computational offloading problem in collaborative vehicle networks [105]. To reduce the inferior learning performance resulting from excessive vehicle mobility, an AF-DQN asynchronous federation mechanism was deployed. The simulation results illustrate that the proposed approach can help reduce the total queuing latency and enhance system throughput. To address the task offloading issues in delay restriction vehicular edge computing, [106] suggested two techniques of value-based reinforcement learning: b-FDQO and baseline DQN.

According to a previous study [107], a smart mobility fog agent (MFA) is included in the SDN controller. The authors also developed an adaptive policy for resource allocation that handles offloading tasks along with the user mobility information. It also showed an innovative IoT-fog system based on an SDN controller and a DRL strategy. This system provides awareness of mobility services and responds to environmental changes. In their research study, they also proposed an innovative local search-based DQN strategy that improves system costs (energy and execution time) for the workloads demanded in different time zones. Furthermore, the research paper [108] presents DRL energy-efficient task scheduling and offloading in a fog IoT network based on SDN. A single SDN controller layer was used to centralize network control and orchestration. SDN-fog computing reduces network latency and traffic overhead. In addition, they proposed a DRL technique to enhance latency reduction and minimize energy consumption in dynamic and distributed IoT networks. There have been a number of problems with optimizing task offloading in highly dynamic vehicle networks, including insufficient information, conflicting queuing latency, and high dimensional curse. In [109], a queuing delay-aware task offloading algorithm based on DRL methods was proposed to dynamically improve the task offloading problem and maximize the throughput of user vehicles while meeting the requirements for the long-term queueing delay in collaborative vehicular networks. The simulation results indicate that the proposed method outperforms the D-QLOA and EMM approaches in terms of throughput and end-to-end queuing delay.

3) DOUBLE DQN

A traditional DQN often uses a single mathematical estimator to select and evaluate an action, which could cause the

value of the action to be overestimated [110]. To address this problem, a double DQN (DDQN) was presented in [111]. Specifically, the Double DQN algorithm splits the evaluation and selection processes into two maximum function estimators. The double estimator method does not overestimate the action value, resulting in a more accurate estimation.

A double DQN-based computation offloading policy in a F-RAN considering D2D communication between user devices was proposed in [110]. This study aims to reduce costs by considering both execution delay and power consumption. Simulation findings indicate that the proposed method efficiently minimizes the total cost compared to existing methods. Moreover, in [12], an energy-efficient computational offloading strategy was proposed for a three-layer architecture in IoV, which incorporates layers of cloudlets, RSUs, and fog nodes. In their research, they considered both stationary and moving vehicles as fog servers and developed a DRL method based on queuing theory to coordinate task flows in order to minimize overall energy usage.

For offloading and allocation optimization problems, a double DQN was used to address the high-dimensional state space when there were more wireless devices. A study in [112] suggested a decentralized optimization scheme based on a double DQN called (DOCRRL) for bandwidth allocation and partial offloading. The suggested schema can learn the best way to make decisions when there are strict limits on latency and risk, thereby avoiding the curse of dimensionality caused by a high number of possible actions and states.

4) DUELING DQN

One significant limitation of the DQN algorithm is that, in some states, the value function does not depend on the selected action. The Dueling Deep Q-network (Dueling DQN) algorithm has been proposed as a solution to overcome the aforementioned dilemma [113].

In addition, depending on the network architecture of the DDQN, the agent in the RL gradually acquires a more accurate value. This indicates that the dueling DQN algorithm might achieve a higher performance than the DQN method when it comes to handling problems related to offloading policies and resources. The work in [114] adopts a dueling DQN algorithm to choose the most suitable offloading strategy for each user equipment (UE), which includes offloading to fog access points (FAP), proximity idle UEs, or processing by itself. As the number of UEs requiring offloading increases, the centralized dueling DQN algorithm becomes more complex. Therefore, a preprocessing mechanism is implemented to reduce the complexity of the dueling DQN algorithm by immediately fulfilling some of the UE's task demands. Similarly, [115] proposed a new offloading policy in an F-RAN that uses the dueling DQN method to optimize the overall utility of UEs.

5) DEEP RECURRENT Q-NETWORK (DRQN)

In conventional neural network training, training samples are primarily determined by the current state. However, it performs poorly when processing data with long-term dependency. To overcome this problem, an LSTM algorithm was introduced. The LSTM is more efficient than traditional neural networks in terms of storing and retrieving information. Therefore, it has recently been used in several fields related to sequence processing [116]. The DRQN method enhances the DQN by using an LSTM instead of a fully connected layer [117]. To efficiently memorize the action that maximizes the reward in each state, the DRQN uses LSTM to store additional information regarding action choices within a state. By combining information over a longer period, the DRQN further improves the network's ability to deal with partially observable models [118].

As an alternative to the DQN, a DRQN approach can be used to estimate optimal value functions and effectively deal with inadequate state observations while dealing with partial observability and a large state space. Therefore, many fog computing computational task offloading problems are formed as partially observable Markov decision processes (POMDPs) because the agent can only view a portion of the entire observation environment and academic researchers tackle these issues based on the DRQN method. The DRQN method combines the past state with the current input state, allowing appropriate decisions, even if the present state of the environment is not observable. For instance, the authors of [119] formulated dynamic computation offloading in IoT fog systems as a POMDP and solved it using a DRQN to provide the IoT device's ideal offloading policy for each state. Compared to benchmark offloading techniques, the suggested offloading strategy may efficiently minimize the energy consumption of IoT devices and satisfy the processing delay requirements of computing tasks.

Similarly, a computational offloading and task scheduling method for minimizing energy consumption under delay constraints was proposed by the authors of [120]. The DRQN is then used to address partial observability based on limited data. The results demonstrate that the proposed offloading algorithm outperforms conventional methods. Furthermore, to ensure a specific quality of service for each task and maximize the utilization of resources by collaborating between several fog computing nodes, a joint task offloading strategy and heterogeneous resource allocation using DRQN algorithm was proposed in [121]. The goal of this study was to increase the number of processing tasks within their latency time limits.

Moreover, DRQN enhancements to the DQN algorithm, make it more suitable for handling issues related to the offloading of vehicle tasks in IoV. In [122], an offloading approach using the DRQN algorithm was proposed to reduce the latency of vehicle task offloading. Initially, an actual map was modeled, the task queue was initialized, and a task offloading environment with many service nodes was created. Subsequently, the DQN algorithm, which integrates deep learning with reinforcement learning, was designed to improve the offloading strategy by minimizing offload delay. Finally, because complete information cannot be adequately observed in the environment, the DQN applies

an LSTM model to train its neural network to increase the offloading effectiveness. According to the simulation results, the proposed schema-based offloading of vehicle tasks can significantly minimize vehicle offloading delays.

6) STATE ACTION REWARD STATE ACTION (SARSA)

This is a method for learning about temporal differences in policy. In the on-policy control approach, each state's action is selected while learning how to use a particular policy. The primary difference between SARSA and Q-learning is that, in contrast to Q-learning, modifying the Q-value in the table does not require a maximum reward for the following state. SARSA selects new actions and rewards based on the same policy as the initial action [80]. The study in [123] proposed a fuzzy reinforcement learning (FRL) approach for an energy-saving task offloading schema in a VFC. The FRL combines fuzzy logic with the SARSA method to reduce both the power consumption and response time.

7) COMBING RL OR DRL METHODS

Many researchers have used two RL or DRL methods to address the offloading problems in fog computing. For example, in research [124], the co-offloading of both computation and traffic for industrial applications in fog computing was studied. Initially, they created a table that stored task properties based on a content-centric design. Subsequently, a strategy is proposed for offloading network tasks that require high volume and expensive computation in fog computing, which considers both the computational workload and traffic volume. The author presented a novel formulation that considers both resource constraints and vehicle mobility with the goal of meeting traffic vehicle offloading requirements. Subsequently, the tradeoff between service latency and energy usage was addressed in edge offloading by concentrating on execution usage and response latency. The author solved the suggested cost reduction challenge by devising two RL-based algorithms: the dynamic RLscheduling (DRLS) method and the deep dynamic scheduling (DDS) method. The authors proposed DDS based on DQN and Double DQN. The results demonstrate that the proposed offloading method reduces service delay and energy consumption in comparison to the baseline offloading schemes.

An effective decision-making system was described in [125], which gives fog nodes the intelligence to choose an appropriate algorithm for processing data. By combining reinforcement learning algorithms (SARSA and Q learning) into its architecture, Devote can adapt to the dynamic environment of IoT. The selection of an appropriate algorithm is based on the characteristics of the data, which can be normal, critical, or too critical. To offload crucial data, researchers have proposed a secretary-based online algorithm to selec the most suitable fog node. The mobility of fog nodes and IoT devices is typical in various applications such as vehicle networks. However, the mobility aspect is ignored in this study.

B. POLICY-BASED ALGORITHMS

The policy-based approach aims to discover the best policy to optimize the reward in the future without relying on the value function [80]. This method explicitly builds a policy (mapping π:s→a) representation and maintains it in the memory during the learning process. Policy gradients (PG) and proximal policy optimization(PPO) are typical policy-based RL algorithms. The policy-based approach consists of two primary types of policies: deterministic and stochastic. In deterministic terms, policy (π) produces the same action in all states, whereas under a stochastic policy, the produced action is determined by probability. The advantage of policy-based approaches is that they have superior convergence properties and are efficient in continuous action spaces or in high dimensionality.

1) POLICY GRADIENT (PG)

The policy gradient approach optimizes parameterized policies in relation to expected returns by using gradient descent (long-term cumulative reward). They are not burdened by many of the issues that plague traditional reinforcement learning approaches, such as the lack of assurance of a value function, complexity imposed by continuous states and actions, and intractability issue imposed by unknown state information [126]. In [127], the authors addressed the issue of computation offloading in an Internet of Things (IoT) fog system enabled by energy harvesting (EH). The researchers modeled the issue as a distributed partially observable Markov decision process (Dec-POMDP). Dec-POMDP is designed to maximize the latency-satisfied utility of all the IoT devices. Then, Lagrangian and policy gradient methods were used to find a local optimal solution to the given optimization problem.

2) ADVANTAGE ACTOR-CRITIC (A2C)

The actor–critic deep reinforcement learning method was first introduced in [128] to incorporate the core concept of value-based and policy-based algorithms and concurrently predict two sets of parameters. A2C requires the deployment of actors and critical networks. The actor is responsible for mapping states to actions, whereas the critic is responsible for mapping state–action pairs to the expected cumulative long-term reward.

Using the advantage actor-critic (A2C) algorithm in deep reinforcement learning (DRL), the author of [129] presented a joint optimization method for resource allocation and offloading strategies to decrease the delay for computational tasks in a fog environment. Multiple action dimensions make the network convergence challenging. As a result, this study employs a multi-agent strategy to obtain the solution by dividing the entire offload decision action into multiple sub-actions. Besides, the paper [130] presents a collaborative approach for content caching, radio resource allocation, and computing offloading in fog-enabled IoT, with the goal of reducing the overall delay for all service requests. They then used a model-free reinforcement learning framework to interact with the environment and choose the

best course of action, which eliminated the requirement for a complete system evolution model. As a function approximator, a DNN is used to estimate the value functions owing to the large number of possible system states and actions. The study proposed an actor-critic deep RL method to learn the optimal stochastic policy for radio resource allocation, content caching, and computing offloading. With the help of a critic, the actor represents a stochastically parameterized policy with another DNN and improves the policy.

3) PROXIMAL POLICY OPTIMIZATION (PPO)

The PPO is a deep reinforcement learning method based on the A2C technique. PPO is an extension of the trust region policy optimization (TRPO) approach, which has been successful [131], [132]. The TRPO incorporates a novel objective function called the surrogate objective and a KL-divergence constraint known as a trusted region to enhance the efficiency of other previous policy gradient-based reinforcement learning approaches. Because sudden changes in the policy can result in inconsistent performance, the trust region prevents TRPO from altering the decision-making policy too frequently with each training period update. However, computing the derivative of the KL-divergence is difficult. By introducing a clipped substitute objective function, PPO overcame this TRPO issue. Because the objective function's clipping restricts the policy's ability to alter, the PPO can be trained without a trusted region, which uses fewer computations than TRPO. Compared to other DRL algorithms, the DQN method as an example, performs well on the problem of continuous action. In terms of sample complexity, it also performs better than A2C and is similar to ACER, although it is much simpler. The PPO can also be expanded by adding more workers to accelerate the training.

The study [133] formulated the challenge of assigning limited fog resources to vehicular applications with the goal of minimizing service delays using parked vehicles. Then, a heuristic technique is presented to efficiently find solutions to the problem. To further improve resource allocation, the proposed method is combined with the PPO's utilized data on parking status and vehicle movement in the city's smart environment. Moreover, [134] formulated an unmanned aerial vehicles (UAV)-assisted offloading problem for IoT to jointly reduce the queue length and energy consumption in smart buildings and environments. The control decisions include evaluating whether IoT device tasks should be offloaded or processed locally as well as allocating time resources and bandwidth to IoT devices attached to the UAV. This was reformulated as an MDP-based offloading problem. To achieve a balance between the two conflicting optimization objectives of queue length and energy usage, the author incorporated both objectives into a single reward function. The UTO approach-based PPO was designed to address the issue of UAV-assisted offloading. The proposed method can effectively solve problems faced by high-dimensional consecutive state and action spaces. The proposed strategy successfully addresses the challenges faced by high-dimensional sequential states and action spaces. Besides, [135] developed IIoT blockchain networks and highlighted the use of RL approaches in IIoT blockchain networks such as PPO.

C. HYBRID ALGORITHMS

Value-based and policy-based algorithms were combined into a hybrid algorithm. Their objective was to implement policy-based algorithms to model the policy function, whereas updates to these policy functions were based on value-based algorithms. The DDPG and SAC are typical hybrid algorithms.

1) DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

DDPG can be viewed as a combination of DQN and DPG algorithms [136]. The primary network, target network, and reply memory are the essential components of DDPG networks. Both the primary and target networks consist of two neural networks: actor and critic. The actor network was used to investigate policies, whereas the critic network was used to evaluate the policies. The critic network also provides critical value for enhancing the policy gradient. The state-action pairs, associated rewards, and subsequent states are all stored in replay memory. These samples were chosen randomly by the agent during training to minimize the impact of the data correlation.

In the literature, the DDPG approach has been utilized to tackle several optimization problems related to offloading computations in fog. For example, a fog-based vehicle architecture with computational offloading and service caching was developed in [137]. Peer-pool and fog-pool computation cooperation were used in this design to improve efficiency. To minimize long-term energy utilization and task processing time, an optimization problem is defined for combining computation offloading with service caching. The DDPG algorithm was used to find an optimal solution to the optimization problem. In addition, [138] presented an intelligent computation offloading technique with resources to enhance the cooperative processing efficiency of fog nodes and improve the user service experience. First, the authors of the papers formulated an energy consumption reduction issue for all computer workloads that considers offloading decisions, transmission power, and bandwidth resources. Moreover, a (D3PG-ICO) algorithm based on DDPG is proposed to solve the formulated problem. The recommended technique creates two separate critic networks to better construct the best global computation offloading policy, and a discretization operation that includes continuous variables is incorporated to improve the randomness of the policy exploration. Finally, the simulation results demonstrate that the proposed system is reliable and that the policy for reducing power consumption by offloading computations can be implemented quickly and flexibly.

In addition, a learning-based mobile fog scheme to offload code blocks in IoT was proposed in a research article [142], that maximizes the use of idle edge computing and storage resources. Offloading state sets are modeled as Markov

decision processes (MDPs) and the DDPG algorithm to address the problem of state space explosion and identify an optimal policy for offloading code blocks in different contexts. Besides, [139] investigated the problem of joint task partitioning and power control in a fog computing network with various mobile devices (MDs) and fog devices (FDs), where each MD must accomplish a periodic computing task under the restrictions of latency and power consumption. To satisfy the latency and power consumption restrictions, the tasks of each MD can be divided into subtasks and performed in collaboration with the FDs. To satisfy the energy consumption and delay limitations, the tasks of each MD can be divided into subtasks simultaneously performed by the MD and the FDs. The solution to this issue was a task offloading algorithm based on MADDPG, which determines the task partitioning and transmission power technique for each MD to optimize the performance of the system in terms of power consumption and processing delay. Finally, the study [140] offers a resource allocation and task offloading system based on blockchain. The authors began by examining both direct and indirect trust using a subjective logical aggregation methodology and distributed trust assessment method. The researchers examined different quality of service features and created a smart contract that uses a DRL algorithm called DDPG to maximize fog revenue while meeting the maximum number of user requests. Blockchain facilitates the entire procedure, from task generation to result computation, and all task transactions are recorded in a secure, unchangeable, tamper-resistant ledger

2) SOFT ACTOR-CRITIC (SAC)

SAC is a DRL algorithm based on an off-policy A2C model that provides sample-efficient learning while preserving the advantages of entropy and stability optimization [144], [145]. The primary principle behind SAC is to utilize all available actions to optimize the actor's entropy and expected reward while ensuring task success. This can be accomplished by integrating entropy maximization with the goal of a stochastic actor. SAC is more suitable than other value-based DRL algorithms, DQN and double DQN as examples, for the allocation of vehicular tasks under fluctuating traffic volumes because it is more effective at handling problems with high-dimensional action space. In addition, the SAC can investigate more effective solutions by adding a policy entropy measure to the reward structure. If many optimal options are available, the SAC strategy selects each option with equal probability. Compared with other policy-based DRL algorithms, such as A3C and DDPG, SAC is more robust and sample-efficient, making it simpler to implement improvements in a stochastic vehicular environment. For example, the work in [13] proposed a DRL method based on SAC to handle the task allocation problem to adjust the offloading policy to changes in a dynamic vehicular environment. The proposed approach becomes more robust and generalized by integrating the policy entropy measure into the reward. Based on the simulation results, the proposed scheme ensured that high-priority tasks were performed first. In terms of offloading delay and task completion rate, it outperformed conventional algorithms. Similarly, astudy [142] suggested a SAC-based DRL algorithm to address the issue of V2V partial computation offloading in vehicular fog computing, in which vehicles with inadequate computing capabilities can offload part of their offloading tasks to nearby vehicles with idle computing resources.

In [143] the problem of joint energy and task offloading in UAV-aided 6G intelligent edge networks was discussed, where each UAV can offload part of its energy and task to the FCNs to decrease the amount of power used locally and the time it takes to complete the entire task by implementing SWIPT. The authors formulated the problem as a cooperative multi-agent Markov game for UAVs with the objective of maximizing the total system utility by optimizing the power allocation strategies and task partitioning of each UAV with respect to energy consumption, task size, and average latency during the execution of tasks, taking into account the network dynamics. A multi-agent soft actor–critic (MASAC) approach is proposed to solve the problem formulation.

To sum up, the core difference between the value-based and the policy-based approaches is that in the value-based approaches approaches use temporal difference (TD) learning to approximate the value function rather than learning the policy explicitly [78]. In contrast, the policy-based approaches learn the 'policy' directly without relying on the value function [80]. This method explicitly builds a policy (mapping π:s$\rightarrow$a) representation and maintains it in the memory during the learning process. Another distinction between the value-based approaches and the policy-based approaches is that value-based approaches are suitable for scenarios with small and discrete action-space. Whereas policy -based approaches are more suited for scenarios with large and continuous action spaces. The advantages of policy-based approaches over value-based approaches are that they have superior convergence properties and are efficient in continuous action spaces or high dimensionality [146]. The value-based reinforcement method has the advantages of simplicity and efficiency. On the other hand, policy-based evaluation has the disadvantage of being inefficient and having a high variance. Since both categories have their own advantages and disadvantages, Therefore, hybrid approaches work based on the combination of value-based and policy-based algorithms that take advantage of both categories.

VI. ANALYTICAL DISCUSSION

Existing research utilizing RL or DRL techniques for handling offloading problems in fog computing has been critically analyzed according to the research questions highlighted in section IV-A. The discussion is provided as follows.

Fig. 5, a two-level pie chart depicts the percentages of research articles from the various conferences and journals of different publishers that were studied in this comprehensive review of the literature. The pie chart's inner circle demonstrates that 72% of the research publications

TABLE 4. Side-by-side comparison RL and DRL methods for solving offloading problems in fog.

RL+DRL methods	Ref.	Probem formulation	Performance metric	Utilized Tools	Case Study	Offloading—~~flow~~ direction	Offloading mode	Integrated with SDN	Offloading Decisions	Advantages	Limitations
Q learning	[10]	MDP	Latency	Matlab	General	Vertical (IoT-fog-cloud) Horizontal (fog-fog)	Binary	×	Where/when	• Reducing delay • Load balancing	• No mobility • No security • Cost of communicating and time required to make a decision
	[11]	MDP	Latency	--	General	Vertical (IoT-fog) Horizontal (Fog-Fog)	Binary	✓	What/Where/when	• Reducing latency • Load balancing. • Scalability	• Single point of failure • No security
	[83]	Mathematical	Latency	Pycharm	General	Vertical (IoT-fog) Vertical (IoT - cloud)	Partial(static partitioning)	×	What/Where/when	• low latency	• No energy consumption • without considering cost of communication between edge devices and edge servers • No security • No mobility
	[84]	MDP	Latency, Resource utilization. Scalability, Security	Mininet-Wifi/ Python-based scripts	Vehicular network	Vertical (IoT-to-fog) Horizontal (fog-fog)	Binary	✓	Where/when/How	• Minimizes latency • Utilizing the resources efficiently • Scalability • Security • Load balancing • Low energy • Low latency	• Single point of failure • No mobility
	[85]	MDP	Latency, Energy consumption	NS-3 simulator	Smart city applications	Vertical (IoT - fog-cloud)	Partial(static partitioning)	×	What/Where/when	• Offloading system is secure and effective for balancing energy consumption and latency.	• Small scale environment
	[86]	MDP	Latency, Energy consumption ,Resource utilization	iFogSim	Smart mobile application	Vertical (IoT-fog) Vertical (IoT- - cloud)	Binary	×	Where/when	• Increases the CPU utilization • Reduces the execution time • Decrease energy consumption	• No scalability • No security
	[87]	MDP	Latency, computation costs	-	UAV	Vertical (IoT-fog-cloud)	Binary	×	Where/when/How	• Low computation cost • Low latency	• Small testbed • No security
	[88]	MDP	Energy consumption	-	F-RAN	Vertical (IoT-fog)	Binary	×	Where/when/How	• Reducing energy consumption	• Did not consider latency • No security
	[89]	--	Latency, Energy consumption	NS-3/ C++	Smart factories	Vertical (IoT-fog-cloud)	Binary	×	Where	• Enhance energy effectiveness • Low delay in smart factories	• No security
	[90]	MDP	Latency, Energy consumption	Python	Green industrial	Vertical (IoT-fog-cloud)	Binary	×	Where/when	• Energy wastage • Delay minimization	• No security
	[91]	MDP	Latency, Resource utilization, Scalability	-	Vehicular network	Horizontal (fog-to-fog) Vertical (IoT-fog-cloud)	Binary	✓	Where	• Minimizing the overall latency of time critical IoV applications • Optimizing resource usage in the fog layer • Distributing the IoV tasks optimally	• No security • Single point of failure
	[92]	MDP	Latency, Energy consumption, Resource utilization	iFogsim	Smart surveillance application	Vertical (IoT-fog-cloud)	Partial(dynamic partitioning)	×	What/Where/when	• Low latency • Low Energy • Execution time • Resource utilization	• No Security •
	[93]	-	Latency, Resource utilization, Reliability	PureEdgeSim	Smart city	Vertical (IoT-fog)	Binary	×	Where	• Resource utilization • Low delay	• The technique has a single point of failure; if the orchestrator node fails, the entire system will fail.

on offloading mechanisms based on RL+DRL algorithms in the Fog paradigm are published in journals, and 28% are presented at conferences. The outer circle depicts the many publishers considered for publication of the research

TABLE 4. *(Continued.)* **Side-by-side comparison RL and DRL methods for solving offloading problems in fog.**

DQN	[94]	MDP	Latency	Tensor Flow	F-RAN	Vertical (IoT-fog) Vertical (IoT-cloud)	Binary	×	Where/when/How	• Minimizes latency • Increases throughput	• No mobility • No security
	[95]	-	Energy consumption	-	Industrial Applications	Vertical (IoT-fog) Or Vertical (IoT- -cloud)	Binary	×	Where/when/How	• Minimizing long-term system energy consumption.	• No mobility • No security
	[96]	---	Latency	Tensor Flow	Vehicular network	Horizontal (IoT-IoT) Vertical (IoT-fog)	Binary	×	Where/when/How	• Avoid decision collisions in multi-vehicles task offloading • Reduce the task processing	• No mobility • No security
	[97]	MDP	Latency, Energy consumption, Throughput	UrbanTraffic (SUMO) / MATLAB	Vehicular network	Vertical (IoT-fog) Horizontal (fog-fog)	Binary	✓	Where/when	• Low energy • Low latency • Increase throughput	• No mobility • No security
	[98]	MDP	Latency	-	General	Vertical (IoT-fog)	Binary	×	Where	• Low latency	• No energy consumption • No security • Small scale area
	[99]	MDP	Response time, Energy consumption	-	Healthcare	Vertical (IoT-fog-cloud)	Binary	×	Where/when	• optimize the response time • Enhance energy	• No security
	[100]	MDP	Latency, Throughput	-	Vehicular network	Vertical (IoT-fog)	Binary	×	Where/when/How	• Reduce end-to-end queuing Delay • Increase throughput	• No mobility • No security
	[101]	--	QoE	-	Vehicular network	Vertical (IoT-fog)	Binary	×	Where/How	• Improved QoE with delay constraint of task	• Did not consider both operation cost and energy consumption.
	[102]	MDP	Latency, Energy consumption	Python	Green industrial	Horizontal (fog-to-fog) vertical(fog-to-cloud)	Partial(dynamic partitioning)	×	What/Where/When	• Minimizing processing delay • Energy consumption • Large-scale area • Low energy	• No security •
	[103]	Non-linear problem	Energy consumption	-	Vehicular network	Horizontal (IoT-IoT) Vertical (IoT-fog)	Binary	×	Where/When/How	• Considered the dynamics of mobile vehicular networks	• No security
	[104]	MDP	Latency, Energy consumption	Matlab	Vehicular network	Horizontal (IoT-IoT) Vertical (IoT-fog)	Binary	×	Where/When/How	• Low latency • Low energy	• No energy consumption • Without considering the cost of communication between edge devices and edge servers, • No security • No Mobility
	[14]	Non-linear problem	Latency	Matlab	F-RAN	Vertical (IoT-fog) Or Vertical (IoT- -cloud)	Binary	×	Where/When	• Minimizing system latency	• No security • No scalability in the case of fast dynamic movement and a large number of users
	[105]	MDP	Latency, Throughput	-	Vehicular network	Horizontal (IoT-IoT) Vertical (IoT-fog)	Binary	×	Where/when	• Maximizing the throughput • Reduces end-to-end queuing delay mobility	• No security
	[106]	MDP	QoS	-	Vehicular network	Vertical (IoT-fog)	Binary	×	Where/When	• The scheme has high average QoE • Considered mobility of the vehicles	• No security
	[107]	MDP	Latency, Energy consumption	edgex mobile foundry	IoT application inside mobile devices	Vertical (IoT - fog-cloud)	Binary	✓	Where/When	• Reducing execution time • Low energy consumption	• Security was not taken into account in the mobility network when data was sent from one node in the fog cloud network to another. • Data tempering, validation, and immutability are widely ignored .

TABLE 4. *(Continued.)* **Side-by-side comparison RL and DRL methods for solving offloading problems in fog.**

	[108]	-	Latency, Energy consumption	Mininet	General	Vertical (IoT-fog)	Binary	✓	Where/when/how	• Minimizes network latency • Increased energy efficiency • Scalability	• The mobility of the terminal devices was not considered. • No security
	[109]	-	Latency, Throughput	-	Vehicular network	Horizontal (IoT-IoT) Vertical (IoT-fog)	Binary	×	Where/When	• Low queuing delay • Low throughput	• No mobility • No security
Double DQN	[12]	MDP	Energy consumption	TensorFlow	Vehicular network	Vertical (IoT-fog) Horizontal(fog-fog)	Binary	×	Where/When/How	• Decrease average energy consumption	• Scalability • No security
	[110]	-	Latency, Energy consumption	TensorFlow	F-RAN	Vertical (IoT-fog) Horizontal(IoT-IoT)	Binary	×	Where/When	• Low energy • Low latency	• No security
	[112]	MDP	Latency, Energy consumption, Privacy and security	Pycharm with Tensorflow	General	Vertical (IoT-fog)	Partial(dynamic partitioning)	×	What/Where/when	• Privacy and security cost • Optimized execution delays • save bandwidth, • Low energy consumption • Minimize computation time • Tackle the curse of dimensionality caused by large number of wireless devices	• No mobility
Dueling DQN	[114]	MDP	Latency, Energy consumption	Tensorflow	F-RAN	Vertical (IoT-fog-cloud)/ Horizontal(IoT-IoT)	Binary	×	Where/When/How	• Low energy • Low latency	• No mobility • No security
DRQN	[115]	--	Latency, Energy consumption	-	F-RAN	Vertical (IoT-fog-cloud)/ Horizontal(IoT-IoT)	Binary	×	Where/When/How	• Low energy • Low latency • Ensuring processing latency requirements for computing tasks. • Minimize the energy usage of the IoT device	• No mobility • No security
	[119]	POMDP	Energy consumption	PyTorch	General	Vertical (IoT-fog)	Binary	×	Where/When/How		• No security
	[120]	POMDP)	Latency, Energy consumption	TensorFlow	General	Vertical (IoT-fog-cloud)/ Horizontal(fog-fog)	Partial(static partitioning)	✓	What/Where/When	• Considered heterogeneous task • Minimizing the energy consumption under the corresponding delay constraint of each task. • Cooperation between fog computing nodes with the objective of maximizing resource usage and ensuring the quality of service for each task. • Considered heterogeneous task offloading • Maximize the completion of processing tasks within their latency time restrictions. Scalability	• No security
	[121]	(POMDP	Latency, Resource utilization, QoE	TensorFlow	General	Vertical (IoT-fog-cloud)/ Horizontal(fog-fog)	Partial(static partitioning)	✓	What/Where/when		• No security
SARSA	[122]	POMDP	Latency	PyTorch	Vehicular network	Vertical (IoT-fog)/ Vertical (IoT - cloud)/ Horizontal(IoT-IoT)	Partial(static partitioning)	×	What/Where/When/How	• a lower system energy consumption reduce the tasks offloading latency	• No security • No mobility
	[123]	MDP	Energy consumption	Monte Carlo	Vehicular network	Horizontal(IoT-IoT)	Binary	×	Where	• Minimizing RSU energy within tolerate response time	• No security • No mobility

publications. The majority of these research articles were published by IEEE in both journals (36%) and conferences (28%). On the other hand, 18%, 9%, and 9% were published in Science Direct, Wiley, and Springer respectively.

TABLE 4. *(Continued.)* **Side-by-side comparison RL and DRL methods for solving offloading problems in fog.**

DQN and Double DQN	[124]	MDP	Latency, Energy consumption	TensorFlow ,Matlab	Industrial applications	Vertical (IoT-fog) Horizontal(IoT-IoT)	Binary	×	Where/When	• Trade-off between energy consumption and service delay considers	• Scalability • No security •
SARSA and Q-learning	[125]	MDP	Latency	Matlab	General	Vertical (IoT-fog-cloud)/ Horizontal(fog-fog)	Partial(static partitioning)	×	What/Where/when	• Less service delay	• No mobility • No security
PG											
	[127]	POMDP	Latency, Energy consumption	Matlab	General	Vertical (IoT-fog)	Binary	×	What/Where/when/How	• Guaranteeing IoT device processing latency requirements. • Reduce the energy cost	• No security
	[129]	MDP	Latency, Energy consumption	-	Vehicular network	Vertical (IoT-fog-cloud)	Partial(static partitioning)	×	What/Where/when	• Low energy • Low latency	• No security
A2C	[130]	MDP	latency	-	General	Vertical (IoT-fog) Or Vertical (IoT-cloud)	Binary	×	Where/when/How	• Reduce end-to-end service latency	• No security •
PPO	[133]	-	Latency	Tensorflow	Vehicular network	Vertical (IoT-fog-cloud) Horizontal(fog-fog)	Binary	×	Where/when	• Service latency is minimized.	• No security • Did not consider energy of Vehicles
	[134]	MDP	Energy consumption	-	UAV	Vertical (IoT-fog)	Binary	×	Where/When/ How	• Minimize the queue length • minimize the energy consumption	• No security • No mobility
	[135]	MDP	Latency, Energy consumption, Security	-	General	Vertical (IoT-fog)	Binary	×	Where	• Shortening the delay. • Reducing high power consumption.	• Scalability • No mobility
DDPG	[137]	MDP	Latency, Energy consumption	Tensorflow	Vehicular network	Vertical (IoT-fog-cloud) Horizontal(fog-fog)	Binary	×	Where/when	• Minimize Long-term energy utilization. • Minimize the task processing time	• No security • Did not considering the mobility of the vehicles
	[138]	MDP	Energy consumption	-	General	Vertical (IoT-fog)	Binary	×	Where/when	• Reduce the entire energy consumption of all tasks	• No security and privacy •
	[139]	MDP	Latency	Tensorflow	General	Vertical (IoT-fog-cloud) Horizontal(fog-fog)	Binary	×	Where/When	• Low delay	• Did not focus on the following : • Virtualization • Privacy, Mobility problem of mobile fog with learning based schemes
	[140]	MDP	Latency, Energy consumption	-	General	Vertical (IoT-fog)	Partial(dynamic partitioning)	×	What/Where/when	• Reduce energy consumption • Minimize task execution latency	• The instability during the training process when the number of mobile devices is large • No security
	[141]	MDP	Latency, Energy consumption, security	-	General	Vertical (IoT-fog-cloud)	Binary	×	Where/when	• Trusted offloading schema • reduce task execution delay • Reduce energy consumption	• Did not consider user mobility. • No scalability
SAC	[13]	MDP	Latency	Pytorch	Vehicular network	Vertical (IoT-fog) Horizontal(IoT-IoT)	Partial(static partitioning)	×	What/Where/when	• The mobility of vehicles is taken into account • Optimize the average latency-aware utility of offloading tasks in a period	• No security
	[144]	sequential decision making problem	Latency	-	Vehicular network	Vertical (IoT-fog) Horizontal(IoT-IoT)	Partial(dynamic partitioning)	×	What/Where/when	• Maximizing the mean utility of all the tasks in a task vehicle	• No security • Did not considering the mobility of the vehicles
	[145]	MDP	Latency, Energy consumption	Tensorflow	UAV	Vertical (IoT-fog)	Partial(dynamic partitioning)	×	What/Where/when	• Reduce local energy consumption • Reduce average execution delay • Higher total system utility	• No security

FIGURE 5. Percentage of journals, conferences, and research paper publishers reviewed.

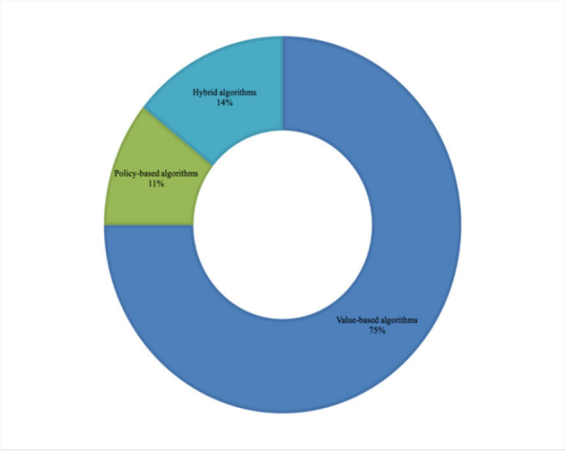

FIGURE 6. Percentage of RL and DRL methods used in fog system offloading based on mode free categorization.

RQ1: What categorization can be applied to RL or DRL based offloading mechanisms in fog computing?

According to the proposed taxonomy, Fig. 6 presents a statistical comparison of the offloading mechanisms based on the RL and DRL approaches for fog. Based on the taxonomy, RL and DRL algorithms are organized into three main categories, namely, value-based algorithms, policy algorithms, and hybrid algorithms As it is illustrated, 62% of the selected articles belong to the value-based approach category. The policy-based approach ranks second with a 29% coverage rate. In addition, only 9% of the reviewed articles were dedicated to hybrid techniques.

RQ2: Which types of algorithms are utilized by RL or DRL-based offloading mechanisms in fog systems?

Fig. 7 shows the percentage of RL and DRL algorithms utilized for solving offloading problems in fog computing. Based on proposed taxonomy and the aforementioned research articles, DQN is the most popular algorithm with 32% usage. As it is illustrated, 21% of the research papers have used Q learning for their proposed offloading schema.

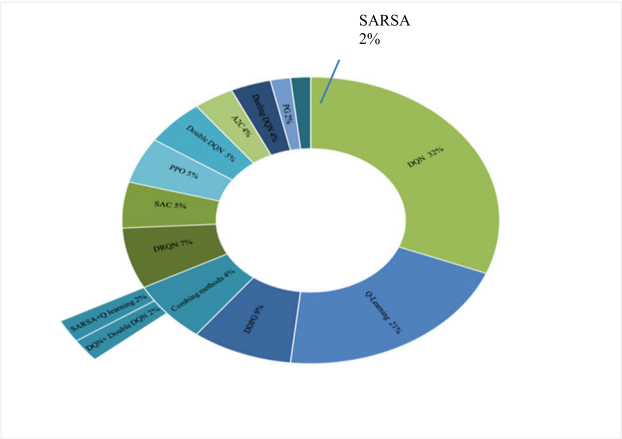

FIGURE 7. Percentage of RL and DRL methods used in fog system offloading mechanism.

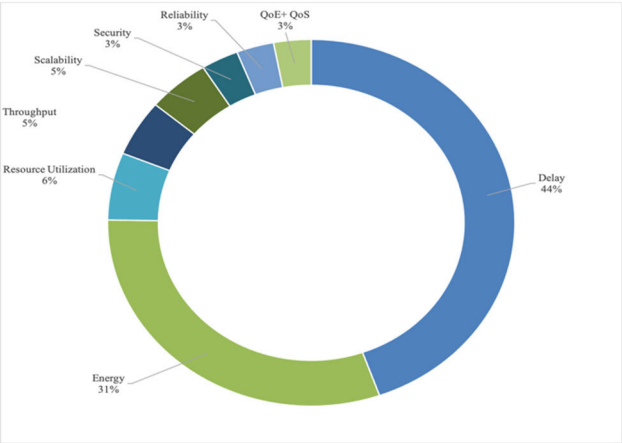

FIGURE 8. Percentage of performance metrics for assessing offloading mechanism.

In addition, 9% and 7% of articles used DDPG and DRQN, respectively. On the other hand, SARSA (2%), PG (2%), and combining methods (SARSA+Q learning (2%), DQN +Double (2%)) are less-used methods. Additionally, the utilized method of offloading mechanisms makes moderate use of the RL and DRL methods: SAC (5%), PPO (5%), double DQN (5%), A2C (4%), and dueling DQN (4%).

RQ3: Which typical performance metrics are utilized in RL or DRL-based offloading techniques in fog computing?

Fig. 8 shows the percentage of performance metrics utilized to assess the offloading mechanisms based on different RL and DRL algorithms. It can be concluded from the figure that latency (46%) and energy (32%), respectively, are two of the most popular metrics used to evaluate offloading strategies. On the other hand, security (3%),QoE+QoS (3%), response time (1%), reliability (1%), and cost (1%) are less-used parameters. Additionally, the evaluation of offloading mechanisms makes moderate use of the following parameters: resource utilization (6%), throughput (5%), and scalability (5%).

RQ4: What cases are studies considered in RL or DRL-based offloading techniques in the fog paradigm?

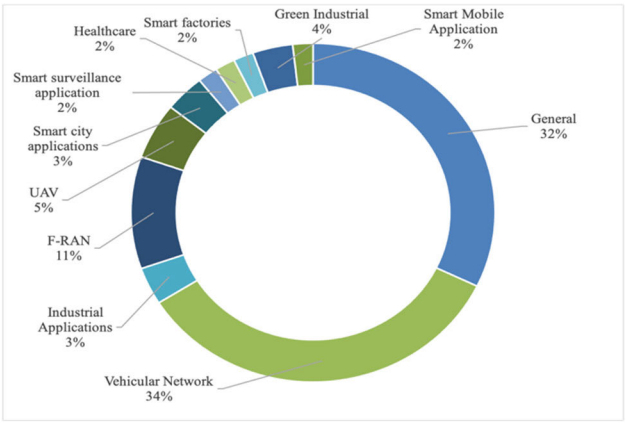

FIGURE 9. Percentage of case study distribution based on offloading mechanism.

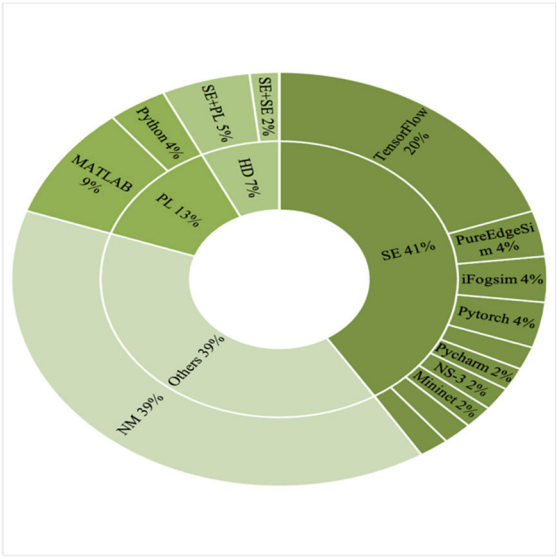

FIGURE 10. Percentage of evaluation tools for evaluating offloading mechanism.

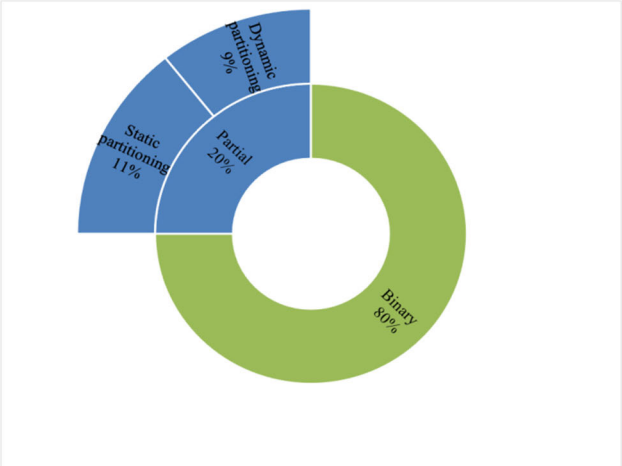

FIGURE 11. Percentage of offloading modes used in fog system offloading mechanism.

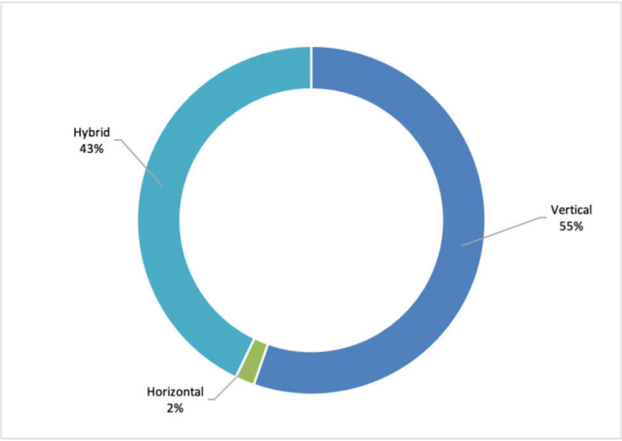

FIGURE 12. Percentage of offloading direction in the fog system offloading mechanism.

Fig. 9 shows the percentage of case studies considered in the RL and DRL approach-based offloading mechanisms in the fog system. The following are examples of existing case studies that were utilized in the experiment results: general apps, vehicular networks, industrial applications, fog radio access (F-RAN), unmanned aerial vehicles (UAV), smart city applications, smart surveillance applications, health care, smart factories, green industry, and smart mobile applications. It concludes from the figure that case studies on vehicular networks, general apps, and F-RANs have been implemented by 34%, 32%, and 11% of the research articles, respectively. The application areas such as UAV(5%), green industrial(4%), smart city application(3%), industrial applications (2%), smart surveillance application(2%), health care(2%), smart factory(2%), and smart mobile application(2%) are rarely explored.

RQ5: Which tools are used to evaluate RL or DRL based mechanisms for offloading in fog systems?

Fig. 10 illustrates several tools utilized for the evaluation of offloading mechanisms based on different RL and DRL methods. The pie chart's inner circle shows the percentage of tools used for evaluating offloading strategies, including simulation environments (41%), programming languages (13%), and hybrid approaches (SE+PL and SE+SE) (7%), and others (39%). In contrast, the outer circle shows that 20% of articles evaluated their proposed approaches using TensorFlow under the simulation category. Besides, PureEdgeSim, iFogSim, and Pytorch simulations have the same percentage of 4%. Other simulation tools, such as Edgex Mobile, Pycharm,NS-3,Mininet, Mininet-Wifi, and Monte-Carlo also have the same percentage of 2%. Furthermore, 9% of the studies used MATLAB, and 4% of the research articles utilized Python, both of which fall under the programming language category. In addition to these, approximately 7% of research papers have used a hybrid approach to assess the various offloading techniques. In about 39% of the publications, no evaluation tool was specified or reported.

RQ6: Which offloading modes are applied in RL or DRL-based offloading techniques in the fog paradigm?

Fig. 11 shows the offloading mode used for the implementation of an offloading mechanism based on different

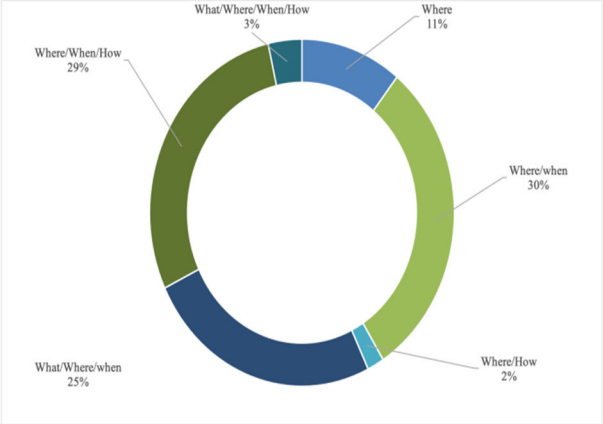

FIGURE 13. Percentage of decisions was made in fog system offloading mechanism.

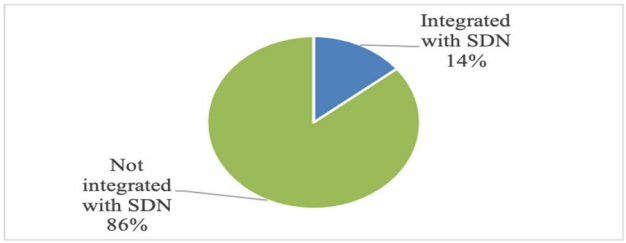

FIGURE 14. The percentage of research articles using or not using SDN in their offloading strategy architecture.

RL and DRL approaches. The pie chart's inner circle shows the percentage of modes used for implementing offloading strategies, including binary offloading (80%) and partial offloading (20%). In contrast, the outer circle shows that, in the partial offloading category, authors of 11% of research articles have used static partitioning to implement their proposed offloading schema, while authors of 9% of research articles have used dynamic partitioning.

RQ7: What offloading direction is usually applied in RL or DRL-based offloading techniques in fog areas?

Fig. 12 shows a statistical comparison of the offloading direction applied in the RL and DRL approach based offloading mechanisms in the fog system. Three offloading directions were considered: vertical, horizontal, and hybrid flows. The vertical offloading mode has the highest percentage of 55% usage in the literature. The hybrid offloading mode, on the other hand, comes in second place, with 43% usage. Finally, with 2% usage, the horizontal offloading mode had the lowest ranking percentage.

RQ8: Which offloading decisions are made with respect to what, where, when, and how are decisions in their strategy based on RL or DRL-based offloading techniques in the fog area?

Fig. 13 shows that the majority of the research articles addressed questions regarding where/when, where/when/how and what/where/when questions by 30%, 29%and 25%, respectively. On the other hand, 11%, 3%, and 2% of decisions were made regarding where, what/where/when/how, and where/how questions.

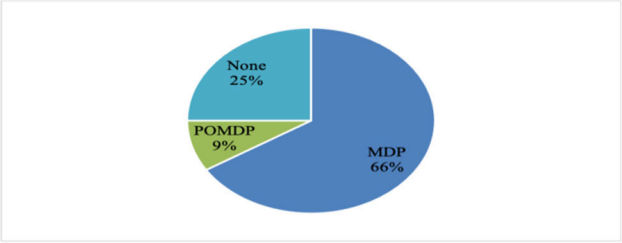

FIGURE 15. Percentage of studies based on offloading problem formulation.

RQ9: Is SDN incorporated into their strategy based on RL or DRL-based offloading mechanisms in fog computing?

Fig. 14 shows that 86% of the reviewed papers did not use SDN in their offloading schema architecture, wheras 14% of the research articles did use SDN in their offloading strategy architecture.

RQ10: How many studies have formulated their offloading strategy problem as an MDP or POMDP problem and solved it based on RL or DRL approaches in a fog environment?

Fig. 15 shows that the RL or DRL method was used in 66% of the research papers to solve the MDP formulation problem. Besides, 9% of the research papers developed their offloading strategy under the POMDP. Also, 25% of research studies used mathematical or non-linear problems to formulate their offloading strategies, or they used RL or DRL methods in their offloading strategies for different purposes, such as finding the shortest route or choosing the best fog candidate.

VII. OPEN ISSUES AND FUTURE TRENDS

This review indicates that some critical issues regarding offloading techniques in fog computing have not yet been researched. To address RQ11, this section discusses several open research issues.

A. SECURITY

One of the significant challenges in modern technology is finding ways to make the system more robust and secure against phishers. When it comes to offloading in fog environments, massive amounts of data are generated from multiple interacting IoT smart devices. This data must be moved to the fog/cloud layer for processing and storage. However, users' devices might mistakenly send their computations or data to neighboring fog servers, which have probably been hacked by a number of attackers, and hence generate serious security concerns. Thus, security issues regularly impact network performance and energy efficiency when multiple overloaded fog and edge nodes are considered. There may be a need for further investigation into smart enterprise security provisioning. It is recommended that blockchain technology be integrated into the proposed strategy to improve the system security.

B. MOBILITY

The offloading scope of a fog environment faces additional obstacles because of its mobility. Important obstacles to

mobility include dynamism and lack of communication. Because mobile devices with naturally dynamic behaviour can move quickly between locations, these devices may need to relocate their dedicated servers across a large geographic area. To solve this issue, a strong mobility management approach must be implemented such that devices can continue to communicate with the fog server even after leaving the source location. Additionally, mobility poses considerable challenges in several research domains, including vehicular networks and unmanned aerial vehicles (UAV). The mobility of the device can impact the offloading performance and cannot be ignored in the stated domains. For instance, a device needs to select a new fog server to offload the task again if it leaves the existing service scope before receiving the offloading result. This results in an increased latency. Therefore, new approaches must be developed to solve these problems efficiently. Despite their significance, mobility difficulties in fog contexts have not received adequate attention in fog environment literature. In addition, fog node selection during offloading depends on the location of the mobile device, which remains a challenging problem.

C. MULTI-OBJECTIVE MECHANISMS

Most existing offloading studies based on reinforcement learning methods in the fog paradigm consider single or bi-objectives in their design. To address the pressing problems in fog computing, including task allocation, job scheduling and offloading, resource provisioning, clustering, cache placement, and load balancing, designing optimization models that jointly optimize multiple objectives (energy consumption, delay, cost, capacity, task execution stability, trust and mobility, and bandwidth reliability) will be an interesting research topic in the future. At the same time, other objectives were overlooked, such as scalability, availability, capacity, security, bandwidth, trust, mobility, and cost. Other relevant obstacles to this unresolved problem include the trade-off between several different objectives, which will become an interesting topic for additional research.

D. IMPLEMENTATION CHALLENGES

It is essential to implement the proposed offloading strategies in real networks to ensure that the QoS requirements are satisfied. However, it is challenging to conduct real-world experiments in a fog computing context because of implementation time and high cost. Therefore, only a few selected publications have validated the proposed offloading mechanisms based on a real-testbed technique. To date, there is no dedicated fog computing testbed to assist researchers in testing their concepts, designs, prototypes, and distributed algorithms in realistic fog computing situations.

E. WORKLOAD PREDICTION IN FOG (OFFLOADING PREDICTION)

Managing resources for various workloads is crucial in fog environments because of the large volume and rapid proliferation of requests. This requires dynamic and efficient resource auto-scaling to optimally distribute fog resources to the requests. A lack of fog resources can lead to a problem known as ''Over-Provisioning'', whereas an abundance of fog resources can lead to ''Under-Provisioning'' [147]. Notably, only a small percentage of the examined studies provided an approach in this regard. Thus, it is essential to study effective auto-scaling to handle fluctuating workloads in fog environments.

F. FINDING OPTIMAL POLICY FOR CANDIDATE FOG SELECTION

Major contributions provide a wide range of network architectures as well as a number of different criteria for action selection. For example, choosing the most powerful computing devices to offload resource-intensive tasks, and choosing the shortest path for offloading. Therefore, finding the best policy that meets multiple goals and is robust is an area of research that has not yet been fully explored and remains open.

G. OFFLOADING MODELING WITH SDN TECHNOLOGY

SDN integrated fog computing is still in its infancy when it comes to offloading computation, and additional research is needed to address several open challenges, particularly in offloading modeling. For both distributed and centralized SDNs, offloading models can be created based on new characteristics such as the timeliness of the collected information.

H. OFFLOADING PARTITIONING

Offloading partitioning refers to the amount of code that must be offloaded and run remotely to increase the effectiveness of resource or time constrained applications in fog computing. There are two different types of computational offloading: binary and partial. Binary offloading is widely used in the literature, whereas partial offloading is weakly covered in the literature. Nevertheless, in practice, task partitioning is essential in some applications, including 3D gaming audio, video, and face recognition, which require more energy and resources than existing devices. Binary offloading is simpler to implement and ideal for simple tasks that cannot be segmented, whereas partial offloading increases the fog server's capabilities but makes offloading modeling considerably complex.

I. OFFLOADING LARGE-SCALE NETWORKS

Fog network offloading techniques for large-scale areas are difficult to solve. The first problem is that a wide network significantly increases the complexity of all offloading model types, which in turn increases the time required to make an offloading decision and the overall offloading delay. Reinforcement learning is a centralized approach that makes it difficult to collect information. The gathering of information on a large-scale network may lead to network

overload and contravene the real-time offloading decision requirements. This is due to the fact that the decision is made after all the necessary information has been gathered, and the amount of time it takes to complete this procedure is not insignificant. In addition, some of the most promising offloading techniques for fog networks must operate efficiently on small-scale networks with a certain number of user devices. However, the reliability of these mechanisms cannot be guaranteed in a large-scale context. Hence, offloading mechanisms on fog networks for large-scale areas will be pursued in the future.

J. HETEROGENEITY AND INTEROPERABILITY

The majority of existing studies on offloading models consider computation tasks to be homogeneous. This makes the modeling of the offloading procedure easier. However, many different tasks are involved in practice. Examples include preemptive and non-preemptive tasks. Modeling is extremely challenging owing to the diversity of tasks. It is also essential to have interoperability, which is the ability of the destination to execute source code correctly. This is required in such heterogeneous computational models to share data and computations. Interoperability also arises in a highly heterogeneous networked environment (consisting of several types of software and hardware from different vendors). In IoT-fog cloud communication, interoperability issues have become increasingly challenging. For example, various cloud storage providers have employed diverse storage technologies. Compression techniques, synchronization systems, and data security and privacy measures may differ for each storage provider. Heterogeneous systems make fog servers more effective. However, they increase the challenge of offloading models. Indeed, the aforementioned open issues provide an opportunity to bring together research based on the offloading mechanisms in fog environments.

K. FAULT TOLERANCE

In addition to security and privacy, fault tolerance is a crucial for establishing trust in task offloading in fog. As discussed in the preceding sections, mobility assistance is one of the most significant requirements during task offloading because freedom of movement and autonomy of communication are essential user satisfaction factors. However, there are several challenges to maintaining continuous connectivity and ongoing access to fog servers during relocation. For instance, data transfer rates and network bandwidth can fluctuate or a connection can be lost. Consequently, task offloading should be improved with fault tolerance mechanisms to ensure that the task is successfully transmitted and executed, as well as to decrease the time energy consumption and application response in end-user devices.

VIII. CONCLUSION

Fog computing has emerged as a promising approach to significantly improve the QoS of user devices and reduce network operational costs by moving computation resources to network edges. Computation offloading is a technique that has the potential to enhance the performance of an application and minimize the overall latency by transferring intensive tasks from resource-limited user devices to a resource-rich fog or cloud server. This paper presents a systematic and comprehensive research analysis of offloading mechanisms based on the RL and DRL methods in a fog computing environment.

By applying our search techniques, 56 research articles were selected for the final selection. According to RQ1, the applied RL or DRL algorithms based on offloading mechanisms in fog computing can be classified into three categories: value-based, policy-based, and hybrid-based algorithms. These categories are then compared based on important features, including offloading problem formulation, utilized techniques, performance metrics, evaluation tools, case studies, their strengths and drawbacks, offloading directions, offloading mode, SDN based architecture, and offloading decisions.

According to RQ2, DQN and Q learning were the most common algorithms used in their proposed offloading schema by 32% and 21%, respectively. Based on RQ3, the most crucial metrics used to assess the offloading mechanisms were latency (46%) and energy (32%). In the manner of RQ4, the majority of case studies were utilized in the experiment results were vehicular networks (34%), and general apps (32%). Furthermore, regarding RQ5, the highest percentage of studies evaluated their proposed approaches using the TensorFlow simulator. According to RQ6, the offloading modes used for the implementation of their offloading schemas were binary offloading (80%) and partial offloading (20%).

In the manner of RQ7, the vertical offloading mode had the highest percentage of 55% in the literature. Regarding to RQ8, the majority of the research articles addressed questions regarding where/when, by 30%. Furthermore, based on RQ9, only 14% of the research articles did use SDN in their offloading strategy architecture. Regarding to RQ10, 66% of the research papers have formulated their offloading strategy problem into an MDP and solved it based on RL or DRL approaches.

In addition, based on RQ11, existing fog offloading techniques have faced a number of unresolved challenges, including security, mobility, multi-objective mechanisms, implementation challenges, workload prediction in fog, finding optimal policy for candidate fog selection, offloading modeling with SDN technology, offloading partitioning, offloading large-scale networks, heterogeneity and interoperability, and fault tolerance. Based on an extensive study, an offloading mechanism based on RL and DRL method taxonomy has been presented, which will help the research community to achieve a better understanding of the offloading mechanism in fog environments.

REFERENCES

[1] A. Botta, W. Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Generat. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[2] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Collaborative vehicular edge computing networks: Architecture design and research challenges," *IEEE Access*, vol. 7, pp. 178942–178952, 2019.

[3] P. Fraga-Lamas, T. M. Fernández-Caramés, O. Blanco-Novoa, and M. A. Vilar-Montesinos, "A review on industrial augmented reality systems for the industry 4.0 shipyard," *IEEE Access*, vol. 6, pp. 13358–13375, 2018.

[4] A. B. De Souza, P. A. L. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. R. Filho, J. N. De Souza, V. Chamola, V. H. C. De Albuquerque, and B. Sikdar, "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 198214–198243, 2020.

[5] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues," *J. Commun. Netw.*, vol. 24, no. 1, pp. 83–98, Feb. 2022.

[6] B. Kar, W. Yahya, Y.-D. Lin, and A. Ali, "A survey on offloading in federated cloud-edge-fog systems with traditional optimization and machine learning," 2022, *arXiv:2202.10628*.

[7] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019.

[8] S. Patil-Karpe, S. H. Brahmananda, and S. Karpe, "Review of resource allocation in fog computing," in *Smart Intelligent Computing and Applications*. Singapore: Springer, 2020, pp. 327–334.

[9] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[10] S. Aljanabi and A. Chalechale, "Improving IoT services using a hybrid fog-cloud offloading," *IEEE Access*, vol. 9, pp. 13775–13788, 2021.

[11] J.-Y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–7.

[12] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. P. C. Rodrigues, X. Kong, J. Huang, and R. Y. K. Kwok, "Deep reinforcement learning for intelligent Internet of Vehicles: An energy-efficient computational offloading scheme," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Jul. 2019.

[13] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.

[14] Z. Chen and X. Su, "Computation offloading and resource allocation based on cell-free radio access network," in *Proc. IEEE 6th Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Mar. 2022, pp. 1498–1502.

[15] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: A comprehensive review," *J. Grid Comput.*, vol. 18, no. 1, pp. 1–42, Mar. 2020.

[16] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective," *Softw., Pract. Exper.*, vol. 50, no. 9, pp. 1719–1759, 2020.

[17] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107496.

[18] A. Shakarami and M. Ghobaei-Arani, "A survey on the computation offloading approaches in mobile edge/cloud computing environment: A stochastic-based perspective," *Comput. Netw.*, vol. 182, Aug. 2020, Art. no. 107496.

[19] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "Journal of network and computer applications a survey on computation offloading modeling for edge computing," *J. Netw. Comput. Appl.*, vol. 169, May 2020, Art. no. 102781.

[20] T. Zheng, J. Wan, J. Zhang, C. Jiang, and G. Jia, "A survey of computation offloading in edge computing," in *Proc. Int. Conf. Comput., Inf. Telecommun. Syst. (CITS)*, Oct. 2020, pp. 1–6.

[21] S. M. Salman, T. A. Sitompul, A. V. Papadopoulos, and T. Nolte, "Fog computing for augmented reality: Trends, challenges and opportunities," in *Proc. IEEE Int. Conf. Fog Comput. (ICFC)*, Apr. 2020, pp. 56–63.

[22] S. S. Hajam and S. A. Sofi, "IoT-fog architectures in smart city applications: A survey," *China Commun.*, vol. 18, no. 11, pp. 117–140, Nov. 2021.

[23] S. Duggal and P. Kaur, "Fog computing based health care applications and frameworks: A review," in *Proc. 8th Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 2021, pp. 238–243.

[24] S. O. Ogundoyin and I. A. Kamil, "Optimization techniques and applications in fog computing: An exhaustive survey," *Swarm Evol. Comput.*, vol. 66, Oct. 2021, Art. no. 100937.

[25] M. Sheikh, S. Mostafa, H. Kashani, and E. Mahdipour, "Towards effective offloading mechanisms in fog computing," *Multimedia Tools Appl.*, vol. 81, pp. 1997–2042, Oct. 2021.

[26] M. Gill and D. Singh, "A comprehensive study of simulation frameworks and research directions in fog computing," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100391.

[27] A. Mekrache, A. Bradai, E. Moulay, and S. Dawaliby, "Deep reinforcement learning techniques for vehicular networks: Recent advances and future trends towards 6G," *Veh. Commun.*, vol. 33, Jan. 2022, Art. no. 100398.

[28] K. Gasmi, S. Dilek, S. Tosun, and S. Ozdemir, "A survey on computation offloading and service placement in fog computing-based IoT," *J. Supercomput.*, vol. 78, no. 2, pp. 1983–2014, Feb. 2021.

[29] D. Alsadie, "Resource management strategies in fog computing environment—A comprehensive review," *Int. J. Comput. Sci. Netw. Secur.*, vol. 22, no. 4, pp. 310–328, 2022.

[30] H. Sabireen and V. Neelanarayanan, "A review on fog computing: Architecture, fog with IoT, algorithms and research challenges," *ICT Exp.*, vol. 7, no. 2, pp. 162–176, Jun. 2021.

[31] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. MCC Workshop Mobile Cloud Comput.*, Aug. 2012, pp. 13–15.

[32] K. S. Awaisi, A. Abbas, M. Zareei, H. A. Khattak, M. U. S. Khan, M. Ali, I. U. Din, and S. Shah, "Towards a fog enabled efficient car parking architecture," *IEEE Access*, vol. 7, pp. 159100–159111, 2019.

[33] Z. Á. Mann, "Notions of architecture in fog computing," *Computing*, vol. 103, no. 1, pp. 51–73, Jan. 2021.

[34] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, 2019.

[35] T.-A.-N. Abdali, R. Hassan, A. H. M. Aman, and Q. N. Nguyen, "Fog computing advancement: Concept, architecture, applications, advantages, and open issues," *IEEE Access*, vol. 9, pp. 75961–75980, 2021.

[36] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.

[37] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *J. Netw. Comput. Appl.*, vol. 98, pp. 27–42, Nov. 2017.

[38] J. Singh, P. Singh, and S. S. Gill, "Fog computing: A taxonomy, systematic review, current trends and research challenges," *J. Parallel Distrib. Comput.*, vol. 157, pp. 56–85, Nov. 2021.

[39] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 1, no. 4, pp. 14–23, Oct. 2009.

[40] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1–5.

[41] M. Babar, M. S. Khan, F. Ali, M. Imran, and M. Shoaib, "Cloudlet computing: Recent advances, taxonomy, and challenges," *IEEE Access*, vol. 9, pp. 29609–29622, 2021.

[42] Y. Ai, M. Peng, and K. Zhang, "Edge cloud computing technologies for Internet of Things: A primer," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 77–86, 2018.

[43] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Comput. Netw.*, vol. 130, pp. 94–120, Jan. 2018.

[44] D. Koustabh and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. Global Internet Things Summit (GIoTS)*, 2017, pp. 1–6.

[45] Y. Xiao, L. Xiao, S. Member, K. Wan, and H. Yang, "Reinforcement learning based energy-efficient collaborative inference for mobile edge computing," *IEEE Trans. Commun.*, early access, Dec. 14, 2022, doi: 10.1109/TCOMM.2022.3229033.

[46] M. Maray and J. Shuja, "Computation offloading in mobile cloud computing and mobile edge computing: Survey, taxonomy, and open issues," *Mobile Inf. Syst.*, vol. 2022, pp. 1–17, Jun. 2022.

[47] W. Tang, X. Zhao, W. Rafique, and W. Dou, "A blockchain-based offloading approach in fog computing environment," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Ubiquitous Comput. Commun., Big Data Cloud Comput., Social Comput. Netw., Sustain. Comput. Commun.*, Dec. 2018, pp. 308–315.

[48] Y. I. Alzoubi, V. H. Osmanaj, A. Jaradat, and A. Al-Ahmad, "Fog computing security and privacy for the Internet of Thing applications: State-of-the-art," *Secur. Privacy*, vol. 4, no. 2, p. e145, Mar. 2021.

[49] Y.-L. Jiang, Y.-S. Chen, S.-W. Yang, and C.-H. Wu, "Energy-efficient task offloading for time-sensitive applications in fog computing," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2930–2941, Sep. 2019.

[50] G. Caiza, M. Saeteros, W. Oñate, and M. V. Garcia, "Fog computing at industrial level, architecture, latency, energy, and security: A review," *Heliyon*, vol. 6, no. 4, Apr. 2020, Art. no. e03706.

[51] M. H. Kashani, A. M. Rahmani, and N. J. Navimipour, "Quality of service-aware approaches in fog computing," *Int. J. Commun. Syst.*, vol. 33, no. 8, p. e4340, May 2020.

[52] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[53] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed Petri nets," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1216–1228, Oct. 2017.

[54] V. Jain and B. Kumar, "Optimal task offloading and resource allotment towards fog-cloud architecture," in *Proc. 11th Int. Conf. Cloud Comput., Data Sci. Eng.*, Jan. 2021, pp. 233–238.

[55] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, and S. Papavassiliou, "Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions," *Comput. Netw.*, vol. 195, Jan. 2021, Art. no. 108177.

[56] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Mobile Netw. Appl.*, vol. 27, no. 3, pp. 1123–1130, Jun. 2022.

[57] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, Feb. 2019.

[58] N. D. Vahed, M. Ghobaei-Arani, and A. Souri, "Multiobjective virtual machine placement mechanisms using nature-inspired Metaheuristic algorithms in cloud environments: A comprehensive review," *Int. J. Commun. Syst.*, vol. 32, no. 14, p. e4068, Sep. 2019.

[59] S. Wu, W. Xia, W. Cui, Q. Chao, Z. Lan, F. Yan, and L. Shen, "An efficient offloading algorithm based on support vector machine for mobile edge computing in vehicular networks," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2018, pp. 1–6.

[60] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Gener. Comput. Syst.*, vol. 87, pp. 278–289, Oct. 2018.

[61] S. Nguyen, Z. Salcic, X. Zhang, and A. Bisht, "A low-cost two-tier fog computing testbed for streaming IoT-based applications," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6928–6939, Apr. 2021.

[62] X. Kui, Y. Sun, S. Zhang, and Y. Li, "Characterizing the capability of vehicular fog computing in large-scale urban environment," *Mobile Netw. Appl.*, vol. 23, no. 4, pp. 1050–1067, Aug. 2018.

[63] N. K. Giang, R. Lea, and V. C. M. Leung, "Developing applications in large scale, dynamic fog computing: A case study," *Softw., Pract. Exper.*, vol. 50, no. 5, pp. 519–532, May 2020.

[64] C. Li, H. Zhuang, Q. Wang, and X. Zhou, "Computer engineering and computer science SSLB: Self-similarity-based load balancing for large-scale fog computing," *Arab. J. Sci. Eng.*, vol. 43, no. 12, pp. 7487–7498, 2018.

[65] Q. Xu and J. Zhang, "PiFogBed: A fog computing testbed based on raspberry pi," in *Proc. IEEE 38th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Oct. 2019, pp. 1–8.

[66] A. Coutinho, F. Greve, C. Prazeres, and J. Cardoso, "Fogbed: A rapid-prototyping emulation environment for fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[67] Y. Karim and R. Hasan, "FogTestBed: A generic architecture for testbed for fog-based systems," in *Proc. SoutheastCon*, Mar. 2020, pp. 1–7.

[68] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exper.*, vol. 47, no. 9, pp. 1–22, 2016.

[69] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[70] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran, "EmuFog: Extensible and scalable emulation of large-scale fog computing infrastructures," in *Proc. IEEE Fog World Congr. (FWC)*, Oct. 2017, pp. 1–6.

[71] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," in *Proc. 2nd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, May 2017, pp. 1–17.

[72] C. Puliafito, D. M. Gonçalves, M. M. Lopes, Leonardo L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencour, "MobFogSim: Simulation of mobility and migration for fog computing," *Simul. Model. Pract. Theory*, vol. 101, May 2019, Art. no. 102062.

[73] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments," *J. Syst. Softw.*, vol. 190, Aug. 2022, Art. no. 111351.

[74] S. Thrun and M. L. Littman, "Reinforcement learning: An introduction," *AI Mag.*, vol. 21, no. 1, p. 103, 2018.

[75] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*.

[76] N. Akalin and A. Loutfi, "Reinforcement learning approaches in social robotics," *Sensors*, vol. 21, no. 4, p. 1292, Feb. 2021.

[77] S. Gupta, G. Singal, and D. Garg, "Deep reinforcement learning techniques in diversified domains: A survey," *Arch. Comput. Methods Eng.*, vol. 28, no. 7, pp. 4715–4754, Dec. 2021.

[78] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends® Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, Nov. 2018.

[79] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, and A. Graves, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[80] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, *arXiv:1708.05866*.

[81] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: An overview," 2017, *arXiv:1806.08894*.

[82] C. J. C. H. Watkins, "Q-learning," *Mach. learning*, vol. 292, pp. 279–292, May 1992.

[83] G. Yang, L. Hou, H. Cheng, X. He, D. He, and S. Chan, "Computation offloading time optimisation via Q-learning in opportunistic edge computing," *IET Commun.*, vol. 14, no. 21, pp. 3898–3906, Dec. 2020.

[84] J. Alotaibi and L. Alazzawi, "SaFIoV: A secure and fast communication in fog-based Internet-of-Vehicles using SDN and blockchain," in *Proc. IEEE Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2021, pp. 334–339.

[85] A. A. Alli and M. M. Alam, "SecOFF-FCIoT: Machine learning based secure offloading in fog-cloud of things for smart city applications," *Internet Things*, vol. 7, Sep. 2019, Art. no. 100070.

[86] A. Shahidinejad and M. Ghobaei-Arani, "Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach," *Softw., Pract. Exper.*, vol. 50, no. 12, pp. 2212–2230, Dec. 2020.

[87] C. Qu, P. Calyam, J. Yu, A. Vandanapu, O. Opeoluwa, K. Gao, S. Wang, R. Chastain, and K. Palaniappan, "DroneCOCoNet: Learning-based edge computation offloading and control networking for drone video analytics," *Future Gener. Comput. Syst.*, vol. 125, pp. 247–262, Dec. 2021.

[88] T. Wei, Y. Sun, Y. Zhang, Z. Wang, W. Wu, and J. Gao, "Energy efficient user access and computation offloading strategy for fog radio access network with uplink/downlink decoupling," in *Proc. IEEE 5th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2019, pp. 894–900.

[89] A. Rafiq, W. Ping, W. Min, and M. S. A. Muthanna, "Fog assisted 6TiSCH tri-layer network architecture for adaptive scheduling and energy-efficient offloading using rank-based Q-learning in smart industries," *IEEE Sensors J.*, vol. 21, no. 22, pp. 25489–25507, Nov. 2021.

[90] A. Hazra and T. Amgoth, "CeCO: Cost-efficient computation offloading of IoT applications in green industrial fog networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6255–6263, Sep. 2022.

[91] M. Ibrar, A. Akbar, S. R. U. Jan, M. A. Jan, L. Wang, H. Song, and N. Shah, "ARTNet: AI-based resource allocation and task offloading in a reconfigurable Internet of Vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 1, pp. 67–77, Jan. 2022.

[92] F. Jazayeri, A. Shahidinejad, and M. Ghobaei-Arani, "Autonomous computation offloading and auto-scaling the in the mobile fog computing: A deep reinforcement learning-based approach," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 8, pp. 8265–8284, Aug. 2021.

[93] Z. Safavifar, S. Ghanadbashi, and F. Golpayegani, "Adaptive workload orchestration in pure edge computing: A reinforcement-learning model," in *Proc. IEEE 33rd Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2021, pp. 856–860.

[94] G. M. S. Rahman, T. Dang, and M. Ahmed, "Deep reinforcement learning based computation offloading and resource allocation for low-latency fog radio access networks," *Intell. Converged Netw.*, vol. 1, no. 3, pp. 243–257, Dec. 2020.

[95] Y. Ren, Y. Sun, and M. Peng, "Deep reinforcement learning based computation offloading in fog enabled industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4978–4987, Jul. 2021.

[96] J. Zhao, M. Kong, Q. Li, and X. Sun, "Contract-based computing resource management via deep reinforcement learning in vehicular fog computing," *IEEE Access*, vol. 8, pp. 3319–3329, 2020.

[97] U. Maan and Y. Chaba, "Deep Q-network based fog node offloading strategy for 5G vehicular adhoc network," *Ad Hoc Netw.*, vol. 120, Sep. 2021, Art. no. 102565.

[98] J. Zong, F. Yang, and X. Luo, "Optimal query policy and task offloading in dynamic environments," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[99] F. Firouzi, B. Farahani, E. Panahi, and M. Barzegari, "Task offloading for edge-fog-cloud interplay in the healthcare Internet of Things (IoT)," in *Proc. IEEE Int. Conf. Omni-Layer Intell. Syst. (COINS)*, Aug. 2021, pp. 1–8.

[100] C. Pan, Z. Wang, Z. Zhou, and X. Ren, "Deep reinforcement learning-based URLLC-aware task offloading in collaborative vehicular networks," *China Commun.*, vol. 18, no. 7, pp. 134–146, Jul. 2021.

[101] D. B. Son, V. T. An, T. T. Hai, B. M. Nguyen, N. P. Le, and H. T. T. Binh, "Fuzzy deep Q-learning task offloading in delay constrained vehicular fog computing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.

[102] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Collaborative AI-enabled intelligent partial service provisioning in green industrial fog networks," *IEEE Internet Things J.*, early access, Sep. 7, 2021, doi: 10.1109/JIOT.2021.3110910.

[103] S. M. A. Kazmi, S. Otoum, R. Hussain, and H. T. Mouftah, "A novel deep reinforcement learning-based approach for task-offloading in vehicular networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 12–17.

[104] H. Tan and L. Zhu, "Overall computing offloading strategy based on deep reinforcement learning in vehicle fog computing," *J. Eng.*, vol. 2020, no. 11, pp. 1080–1087, Nov. 2020.

[105] G. Tian, Y. Ren, C. Pan, Z. Zhou, and X. Wang, "Asynchronous federated learning empowered computation offloading in collaborative vehicular networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2022, pp. 315–320.

[106] D. B. Son, T. H. Binh, H. K. Vo, B. M. Nguyen, H. T. T. Binh, and S. Yu, "Value-based reinforcement learning approaches for task offloading in delay constrained vehicular edge computing," *Eng. Appl. Artif. Intell.*, vol. 113, Aug. 2022, Art. no. 104898.

[107] A. Lakhan, M. A. Mohammed, O. I. Obaid, C. Chakraborty, K. H. Abdulkareem, and S. Kadry, "Efficient deep-reinforcement learning aware resource allocation in SDN-enabled fog paradigm," *Automated Softw. Eng.*, vol. 29, no. 1, pp. 1–25, May 2022.

[108] B. Sellami, A. Hakiri, S. B. Yahia, and P. Berthou, "Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network," *Comput. Netw.*, vol. 210, Jun. 2022, Art. no. 108957.

[109] Z. Jia, Z. Zhou, X. Wang, and S. Mumtaz, "Learning-based queuing delay-aware task offloading in collaborative vehicular networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[110] F. Jiang, X. Zhu, and C. Sun, "Double DQN based computing offloading scheme for fog radio access networks," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Jul. 2021, pp. 1131–1136.

[111] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[112] H. C. Ke, H. Wang, H. W. Zhao, and W. J. Sun, "Deep reinforcement learning-based computation offloading and resource allocation in security-aware mobile edge computing," *Wireless Netw.*, vol. 27, no. 5, pp. 3357–3373, Jul. 2021.

[113] W. Ziyu, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and F. Nando, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[114] F. Jiang, R. Ma, Y. Gao, and Z. Gu, "A reinforcement learning-based computing offloading and resource allocation scheme in F-RAN," *EURASIP J. Adv. Signal Process.*, vol. 2021, no. 1, pp. 1–25, Dec. 2021.

[115] F. Jiang, R. Ma, C. Sun, and Z. Gu, "Dueling deep Q-network learning based computing offloading scheme for F-RAN," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, Aug. 2020, pp. 1–6.

[116] K. Greff, R. K. Srivastava, J. Koutnìk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space Odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[117] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *Proc. AAAI Fall Symp. Ser.*, 2015, p. 23.

[118] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva, "Deep attention recurrent Q-network," 2015, *arXiv:1512.01693*.

[119] R. Xie, Q. Tang, C. Liang, F. R. Yu, and T. Huang, "Dynamic computation offloading in IoT fog systems with imperfect channel-state information: A POMDP approach," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 345–356, Jan. 2021.

[120] J. Baek and G. Kaddoum, "Online partial offloading and task scheduling in SDN-fog networks with deep recurrent reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11578–11589, Jul. 2022.

[121] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1041–1056, Jan. 2021.

[122] T. Wang, X. Luo, and W. Zhao, "Improving the performance of tasks offloading for Internet of Vehicles via deep reinforcement learning methods," *IET Commun.*, vol. 16, no. 10, pp. 1230–1240, Jun. 2022.

[123] S. Vemireddy and R. R. Rout, "Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing," *Comput. Netw.*, vol. 199, Nov. 2021, Art. no. 108463.

[124] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 976–986, Feb. 2019.

[125] M. Tiwari, S. Misra, P. K. Bishoyi, and L. T. Yang, "Devote: Criticality-aware federated service provisioning in fog-based IoT environments," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10631–10638, Jul. 2021.

[126] H. Dong, H. Dong, Z. Ding, and S. Zhang, *Deep Reinforcement Learning*. Singapore: Springer, 2020.

[127] Q. Tang, R. Xie, F. R. Yu, T. Huang, and Y. Liu, "Decentralized computation offloading in IoT fog computing system with energy harvesting: A dec-POMDP approach," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4898–4911, Jun. 2020.

[128] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[129] W. Bai and C. Qian, "Deep reinforcement learning for joint offloading and resource allocation in fog computing," in *Proc. IEEE 12th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Aug. 2021, pp. 131–134.

[130] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.

[131] S. John, S. Levine, P. Abbeel, M. Jordan, and M. Philipp, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[132] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[133] S.-S. Lee and S. Lee, "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10450–10464, Oct. 2020.

[134] J. Xu, D. Li, W. Gu, and Y. Chen, "UAV-assisted task offloading for IoT in smart buildings and environment via deep reinforcement learning," *Building Environ.*, vol. 222, Aug. 2022, Art. no. 109218.

[135] T. Alam, A. Ullah, and M. Benaida, "Deep reinforcement learning approach for computation offloading in blockchain-enabled communications systems," *J. Ambient Intell. Humanized Comput.*, vol. 2022, pp. 1–14, Jan. 2022.
[136] J. J. Hunt, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[137] D. Lan, A. Taherkordi, F. Eliassen, and L. Liu, "Deep reinforcement learning for computation offloading and caching in fog-based vehicular networks," in *Proc. IEEE 17th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Dec. 2020, pp. 622–630.
[138] S. Chen, B. Tang, and K. Wang, "Twin delayed deep deterministic policy gradient-based intelligent computation offloading for IoT," *Digit. Commun. Netw.*, vol. 2022, pp. 1–13, Jun. 2022.
[139] Z. Cheng, M. Min, M. Liwang, L. Huang, and Z. Gao, "Multiagent DDPG-based joint task partitioning and power control in fog computing networks," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 104–116, Jan. 2022.
[140] V. Jain and B. Kumar, "Blockchain enabled trusted task offloading scheme for fog computing: A deep reinforcement learning approach," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 11, Nov. 2022, Art. no. e4587m.
[141] M. Chen, T. Wang, S. Zhang, and A. Liu, "Deep reinforcement learning for computation offloading in mobile edge computing environment," *Comput. Commun.*, vol. 175, pp. 1–12, Jul. 2021.
[142] J. Shi, J. Du, J. Wang, and J. Yuan, "Deep reinforcement learning-based V2V partial computation offloading in vehicular fog computing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2021, pp. 1–6.
[143] Z. Cheng, M. Liwang, N. Chen, L. Huang, X. Du, and M. Guizani, "Deep reinforcement learning-based joint task and energy offloading in UAV-aided 6G intelligent edge networks," *Comput. Commun.*, vol. 192, pp. 234–244, Aug. 2022.
[144] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[145] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.
[146] M. Sewak and R. Learning, "Policy-based reinforcement learning approaches: Stochastic policy gradient and the REINFORCE algorithm," in *Deep Reinforcement Learning*. Singapore: Springer, 2019, pp. 127–140.
[147] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "A cost-efficient auto-scaling mechanism for IoT applications in fog computing environment: A deep learning-based approach," *Cluster Comput.*, vol. 24, no. 4, pp. 3277–3292, Dec. 2021.

DEZHEEN H. ABDULAZEEZ received the B.S. degree in computer science from the University of Duhok, in 2011, and the M.S. degree in web application and services from the University of Leicester, U.K., in 2015. She is currently pursuing the Ph.D. degree in fog computing with the College of Science, University of Duhok. Her research interests include fog computing, the Internet of Thing (IoT), sematic web, and web applications and services.

SHAVAN K. ASKAR received the B.Sc. (Hons.) and M.Sc. degrees from the Control and Systems Engineering Department, Baghdad, in 2001 and 2003, respectively, and the Ph.D. degree in electronic systems engineering from the University of Essex, U.K., in 2012. He is currently the CEO with Arcella Telecom. He is also an Associate Professor of computer networks. He works as an Assistant Professor with the College of Technical Engineering, Erbil Polytechnic University. He also works in the field of networks that includes the Internet of Things, software-defined networks, optical networks, and 5G information systems engineering. His research interests include 5G, the IoT SDN, network virtualization, and fog and cloud computing.