RL/DRL Meets Vehicular Task Offloading Using Edge and Vehicular Cloudlet: A Survey

Article *in* IEEE Internet of Things Journal · June 2022

DOI: 10.1109/JIOT.2022.3155667

8 authors, including:

Manzoor Ahmed
Hubei Engineering University
99 PUBLICATIONS **2,519** CITATIONS
SEE PROFILE

Muhammad Ayzed Mirza
Qilu Institute of Technology
29 PUBLICATIONS **536** CITATIONS
SEE PROFILE

Wali Ullah Khan
University of Luxembourg
221 PUBLICATIONS **4,207** CITATIONS
SEE PROFILE

Dianlei Xu
University of Helsinki
13 PUBLICATIONS **1,272** CITATIONS
SEE PROFILE

RL/DRL Meets Vehicular Task Offloading Using Edge and Vehicular Cloudlet: A Survey

Jinshi Liu, Manzoor Ahmed, Muhammad Ayzed Mirza, Wali Ullah Khan, Dianlei Xu, Jianbo Li, Abdul Aziz ,and Zhu Han , *Fellow, IEEE*

任务卸载可分为两大类：边缘云的RL/DRL解决方案和车辆云的RL/DRL解决方案

***Abstract*—The last two decades have seen a clear trend towards crafting intelligent vehicles based on the significant advances in communication and computing paradigms, which provide a safer, stress-free, and more enjoyable driving experience. Moreover, emerging applications and services necessitate massive volumes of data, real-time data processing, and Ultra-reliable and Low-Latency Communication (URLLC). However, the computing capability of current intelligent vehicles is minimal, making it challenging to meet the delay-sensitive and computation-intensive demand of such applications. In this situation, vehicular task/computation offloading towards the Edge Cloud (EC) and Vehicular Cloudlet (VC) seems to be a promising solution to improve the network's performance and applications' Quality of Service (QoS). At the same time, Artificial Intelligence (AI) has dramatically changed people's lives. Especially for vehicular task offloading applications, AI achieves state-of-the-art performance in various vehicular environments. Motivated by the outstanding performance of integrating Reinforcement Learning (RL)/ Deep Reinforcement Learning (DRL) to the vehicular task offloading systems, we present a survey on various RL/DRL techniques applied to vehicular task offloading. Precisely, we classify the vehicular task offloading works into two main categories: 1) RL/ DRL solutions leveraging edge cloud (EC) and 2) RL/DRL solutions using VC computing. Moreover, the EC section-based RL/DRL solutions are further sub-categorized into Multi-Access Edge Computing (MEC) server, nearby vehicles, and hybrid MEC. As per the authors' knowledge, we are the first to cover RL/DRL-based vehicular task offloading. Also, we provide lessons learned and open research challenges in this field and discuss the possible trend for future research.**

***Index Terms*—AI, machine learning, RL, DRL, vehicular edge and cloudlet computing, task offloading, V2V, V2I, and V2X communications.**

Jinshi Liu, Manzoor Ahmed, and Jianbo Li are with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China (e-mails: jssmith0716@gmail.com, manzoor.achakzai@gmail.com, lijianbo@qdu.edu.cn) Manzoor Ahmed is the corresponding author.

Muhammad Ayzed Mirza is with BUPT-QMUL EM Theory and Application International Research Lab, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: mamirza@bupt.edu.cn).

W. U. Khan is with Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, 4365, Esch-sur-Alzette, Luxembourg (email: waliullah.khan@uni.lu).

Dianlei Xu is with the Department of Computer Science, University of Helsinki, Helsinki, Finland.(e-mails: dianlei.xu@helsinki.fi).

Abdul Aziz is with the Department of Telecommunication Engineering, Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan.(e-mail: abdul.aziz@iub.edu.pk).

Zhu Han is with the University of Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul, South Korea (e-mail:zhan2@uh.edu).

I. Introduction

Nowadays, vehicles are outfitted with a variety of embedded sensors, processing and storage devices, and various kind of wireless communication modules for enabling safe, efficient, and comfortable experience [1]. These vehicular services and applications are continuously evolving and becoming data-hungry, computation-intensive, and delay-sensitive [2], [3]. However, vehicle onboard resources are limited to process computation-intensive or delay-sensitive tasks. For tackling such problems, vehicles need to optimize computation/task offloading and get assistance from other computing paradigms such as Centralized Cloud (CC), Edge-Cloud (EC), and moving/stationary Vehicular Cloudlet (VC) to handle task processing [4]. Although computation offloading is a promising approach to improve the applications' performance and QoS, specifically, it is pretty challenging in a highly dynamic vehicular environment [5], [6].

Most of the Intelligent Transportation Systems (ITS) [7], [8], e.g., the safety/warning applications, aided-driving and Autonomous Vehicle (AV) [9], and infotainment applications and services [10] are computation-intensive. These applications cannot be independently executed by vehicles. These applications, as mentioned earlier, require real-time response. Due to transmission delay, limited backhaul bandwidth, and unstable connections, traditional cloud-based solutions are not feasible. However, EC computing or Multi-access Edge Computing (MEC) servers defined by the European Telecommunication Standard Institute deployed at the roadside infrastructures/Base Stations (BS) seem good architecture. MEC is one of the critical entities in the 5G network and can support innovative services and applications requiring ultra-reliability and low latency. Moreover, for augmenting EC, VC is surfacing as another computing paradigm comprising of a group of vehicles that can be static or dynamic [11]–[13]. To access these computing resources, Vehicle-to-Vehicle (V2V) [14] and Vehicle to Roadside Infrastructure (V2I) [15] communication modes with different Radio Access Technologies (RATs), i.e., Dedicated Short Range Communications (DSRC) and cellular (i.e., LTE and New Radio (NR)) networks could be adopted [16]–[18].

Artificial Intelligence (AI) has continuously evolved and expanded to several domains, including Industry 4.0, the Internet of Things (IoT), and automotive networks. It is expected that 6G wireless communication networks will have imbued with pervasive and distributed AI. The proliferation and deep integration of AI tools with wireless systems can further ameliorate the performance of network functions and decision-making cost-effectively [19]. For example, AI has empowered ITS including traffic flow prediction [20], [21], traffic flow optimization [22], speed prediction [20], traffic accident detection [23], abnormal driving detection [24], [25], distraction detection [26], cooperative lane changing [27], etc. Emerging vehicular applications and services based on AI algorithms, which are mostly delay-sensitive, have enhanced performance compared with conventional algorithms. Specifically, Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) can further enhance the performance of vehicular task offloading [28]. Vehicular task offloading in large-scale complex environments is challenging with traditional algorithms, while RL/DRL algorithms can tackle it effectively.

Considering the complex and highly dynamic vehicular networks due to fast varying channels and computation load in task offloading scenarios, most classic schemes utilizing one-shot optimization may fail to attain stable long-term optimized performance. RL algorithms, e.g., Q-Learning, learning is a good choice; however, they fail to handle large state spaces' eruptions in vehicular task offloading environments. Fortunately, DRL successfully leverages neural networks (NN) as function approximators to extend RL's scalability to adapt to the complex vehicular task offloading environment. In contrast, based on historical experience, the DRL can learn the optimal decision-making policy in real-time [29]. Although DRL's trial-and-error training is a time and resource-consuming process, once converged, then DRL agents can react to the vehicular environment changes in milliseconds to achieve real-time decision making. Furthermore, compared with traditional heuristic algorithms, the experience-driven DRL assumes no environment models. In particular, via the accumulation of new experiences, DRL can improve its policy during offloading operations and adapt to vehicular environmental changes quickly to meet the delay-sensitive applications' requirements.

在利用RL/DRL技术，并结合边缘计算与车辆云来处理车辆任务
卸载问题这一特定领域，还没有一个系统的整理和分类

A. Motivation and Contribution

Several good surveys have focused on vehicular networks, including [30]–[40], but none of them has organized and categorized the vehicular task offloading schemes according to RL/DRL perspective leveraging edge and vehicular clouds. Unlike our proposed survey, the authors in [30] analyzed edge computing. They reviewed how such schemes can improve the performance of vehicular environment IoT networks without paying much attention to vehicular computation offloading. In [31], opportunistic offloading techniques are mainly investigated for mobile data and task offloading; however, the authors briefly discussed vehicular task offloading. By contrast, we focused on vehicular task offloading approaches using RL/DRL tools for making timely and optimal decisions. The authors in [32] studied the Vehicular Edge Computing (VEC) architecture based on three layers, and little attention is given to vehicular task offloading. In contrast, we extensively and deeply cover the task offloading under RL/DRL perspective in a vehicular environment. Similarly, in [33], the authors investigated the VEC and relevant models on computation offloading in detail. In [34], VEC architectures and vehicular communication technologies are targeted with a comprehensive review on the computation offloading approaches. In [35], VEC-based computation offloading, content caching, and delivery approaches were reviewed extensively without focusing on AI approaches. Recently, the authors in [36] put forward a comprehensive survey on computation offloading from the view of communication standards, problems, and experiments used in vehicular environments. More specifically, [36] tackled this survey differently, presenting summary tables in detail, but the research works are not discussed individually. Different from [36], we reviewed the related works on vehicular task offloading leveraging RL/DRL algorithms for EC and VC and then compared each category after dividing it into value-based and policy-based schemes. In [37], stochastic-based offloading approaches in various computation environments, such as Mobile Cloud Computing (MCC), MEC, and Fog Computing, were discussed. The literature is categorized based on the Markov chain, Markov process, and Hidden Markov Models. However, their prime focus was not on vehicular task offloading. In [38], the authors introduced edge intelligence and intelligent edge and discussed their application scenarios, enabling technologies such as DL training and inference in the customized edge computing framework. In [39], the authors presented a MEC-based review on machine learning for computation offloading. The authors considered all the schemes leveraging RL, supervised learning, and unsupervised learning, and provided their comparison based on the essential characteristics of performance indicators, case studies, technologies used, and evaluation tools. However, vehicular task offloading is not focused. In [40], the authors presented a summary of recent federated learning in vehicular networks considering applications, resource management, and performance optimization. Unlike the abovementioned surveys, we deeply investigated vehicular task offloading for edge and vehicular cloudlets, specifically targeting RL/DRL algorithms to enable deep insight into the evolving domain.

Overall, there still lacks an organized survey article keeping complete and solid discussions on RL/DRL-based vehicular task offloading research considering deep integration of EC and VC, which motivates the current work. To fill the gap, we investigate and present a complete and deep study of RL/DRL algorithms for task offloading in a vehicular environment as shown in taxonomy Fig. 1. We categorize the state-of-the-art research works into two domains, i.e., RL/DRL solutions leveraging edge cloud and RL/DRL solutions using vehicular cloudlet. For better insight, different from the earlier surveys, we elaborate the edge cloud and sub-categorize the "RL/DRL solutions leveraging edge cloud" section into MEC server, nearby vehicles, and Hybrid MEC (HMEC). Moreover, the RL/DRL solutions are further

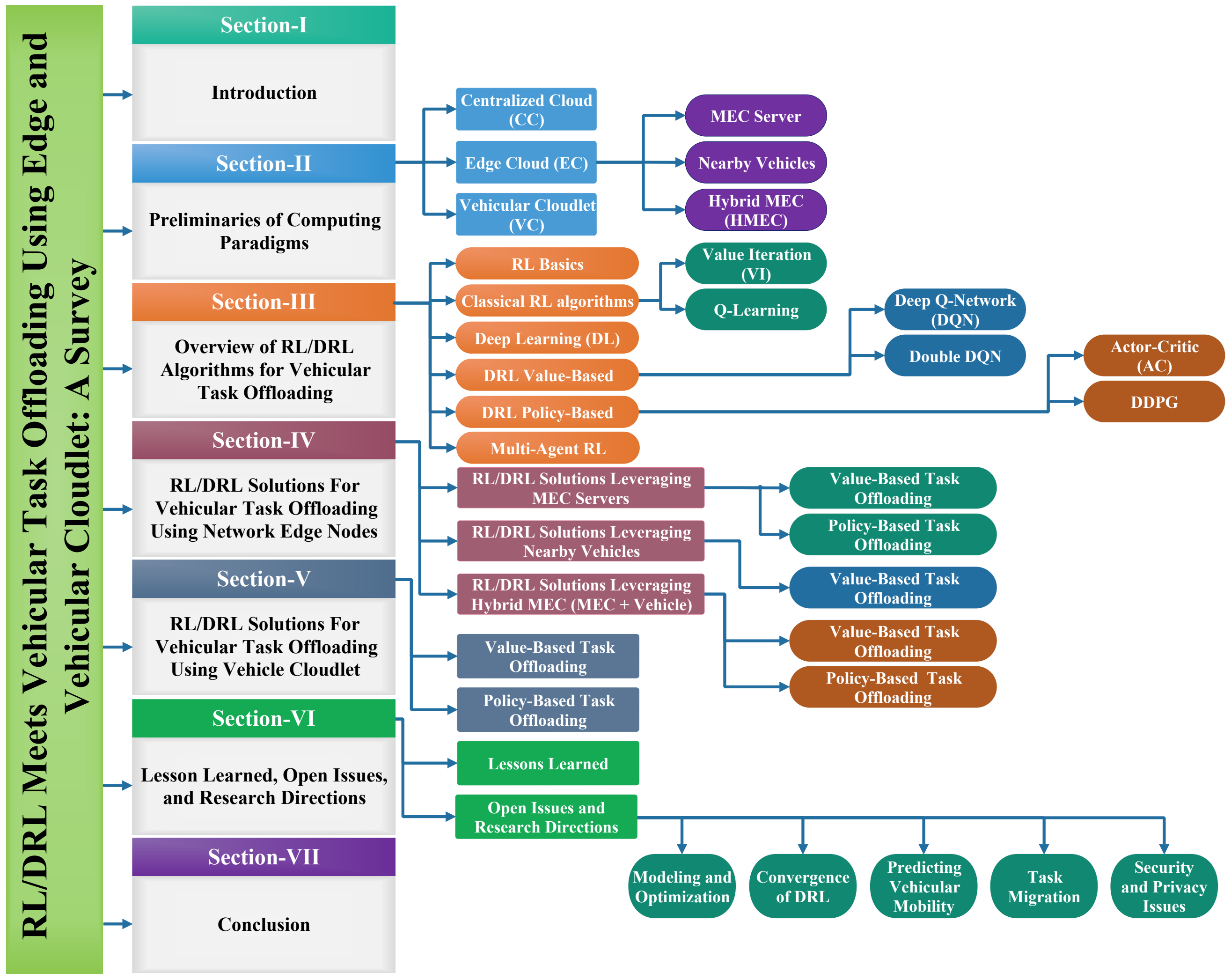
RL/DRL Meets Vehicular Task Offloading Using Edge and Vehicular Cloudlet: A Survey
Section-I
Introduction
Section-II
Preliminaries of Computing Paradigms
Section-III
Overview of RL/DRL Algorithms for Vehicular Task Offloading
Section-IV
RL/DRL Solutions For Vehicular Task Offloading Using Network Edge Nodes
Section-V
RL/DRL Solutions For Vehicular Task Offloading Using Vehicle Cloudlet
Section-VI
Lesson Learned, Open Issues, and Research Directions
Section-VII
Conclusion
Centralized Cloud (CC)
Edge Cloud (EC)
Vehicular Cloudlet (VC)
MEC Server
Nearby Vehicles
Hybrid MEC (HMEC)
RL Basics
Classical RL algorithms
Deep Learning (DL)
DRL Value-Based
DRL Policy-Based
Multi-Agent RL
Value Iteration (VI)
Q-Learning
Deep Q-Network (DQN)
Double DQN
Actor-Critic (AC)
DDPG
RL/DRL Solutions Leveraging MEC Servers
RL/DRL Solutions Leveraging Nearby Vehicles
RL/DRL Solutions Leveraging Hybrid MEC (MEC + Vehicle)
Value-Based Task Offloading
Policy-Based Task Offloading
Value-Based Task Offloading
Value-Based Task Offloading
Policy-Based Task Offloading
Value-Based Task Offloading
Policy-Based Task Offloading
Lessons Learned
Open Issues and Research Directions
Modeling and Optimization
Convergence of DRL
Predicting Vehicular Mobility
Task Migration
Security and Privacy Issues

Fig. 1: Taxonomy of the RL/DRL based vehicular task offloading survey using edge cloud and vehicular cloudlet.

segregated into value-based and policy-based schemes for the edge cloud and vehicular cloudlet sections.

Our article's main contributions are as follows:

- We built upon the available literature on RL/DRL-based vehicular task offloading and organized a comprehensive survey considering the recent advances. We provide a brief highlight of computing paradigms for better understanding, including edge cloud and its variants and vehicular cloudlet. In addition, an overview of RL/DRL algorithms used in vehicular task offloading is given to provide theoretical analysis and offer a quick reference for both naïve and experienced researchers.
- For the RL/DRL-based vehicular task offloading solutions leveraging edge cloud, we elaborate, compare, and analyze the works under each sub-category (i.e., MEC server, nearby vehicles, and HMEC) according to the value-based and policy-based task offloading sub-classes, respectively. Moreover, summary tables are given for each sub-category to get better insight and catch the logical connection of the different task offloading schemes from various aspects such as offloading and algorithm types, RL/DRL elements, and optimization objectives.
- For the RL/DRL-based vehicular task offloading solutions leveraging vehicular cloudlet, we also elaborate, compare, and analyze the research works in this category. Furthermore, to enable the readers to understand the RL/DRL schemes better, a summary table is given considering different facets, i.e., RAT network, algorithms' details, RL/DRL elements, and optimization objectives.
- We provide lessons learned, open issues, and potential research directions for RL/DRL-based vehicular task offloading.

The remaining structure of our paper is as follows. We give a brief overview of computing paradigms and RL/DRL algorithms in Section II and Section III, respectively. In Section IV, we provide the first category of vehicular task offloading, i.e., RL/DRL solutions leveraging edge cloud, followed by its sub-categories, summaries, and discussions.

TABLE I: List of Important Acronyms

Acronym	Definition	Acronym	Definition
A2C	Advantage Actor Critic	NR	New Radio
A3C	Asynchronous Advantage Actor Critic	PER	Priority Experience Replay
AC	Actor Critic	PPO	Proximal Policy Optimization
ACO	Ant Colony Optimization	PSO	Particle Swarm Optimization
ALTO	Adaptive Learning-based Task Offloading	QoS	Quality of Servic
AP	Access Point	RATs	Radio Access Technologies
API	Application Programming Interface	RL	Reinforcement Learning
BS	Base Station	RNN	Recurrent Neural Network
CNN	Convolutional Neural Network	RSUs	Road Side Units
C-V2X	Cellular Vehicle-to-Everything	RU	Resource Unit
DDPG	Deep Deterministic Policy Gradient	SAC	Soft Actor Critic
DDQN	Double Deep Q Network	SDN	Software Defined Network
DL	Deep Learning	SMDP	Semi-Markov Decision Process
DQN	Deep Q Network	TD	Temporal Difference
DRL	Deep Reinforcement Learning	UAV	Unmanned Aerial Vehicles
DSRC	Dedicated Short Range Communication	URLLC	Ultra Reliable and Low Latency Communication
FCFS	First Come First Service	V2I	Vehicle-to-Infrastructure
HMEC	Hybrid MEC	V2V	Vehicle-to-Vehicle
HMM	Hidden Markov Model	V2X	Vehicle-to-Everything
IoT	Internet of Things	VANET	Vehicular ad-hoc Network
KD	Knowledge Driven	VCC	Vehicular Cloud Computing
MBS	Macro-cell Base Station	VCG	Vickrey Clarke Groves
MC	Monte Carlo	VEC	Vehicular Edge Computing
MDP	Markov Decision Process	VFC	Vehicular Fog Computing
MEC	Multi-access Edge Computing	VI	Value Iteration
NN	Neural Network	VM	Virtual Machine

Similarly, in Section V, RL/DRL solutions using vehicular cloudlet computing are given along with the summary and discussion. Next, Section VI discusses the lessons learned, open challenges, and future research directions. Finally, we conclude our paper in Section VII. We also provide the acronyms' definition Table I.

II. Preliminaries of Computing Paradigms

Smart usually refers to the connected vehicles supported by information and communication technologies, which enabled many innovations in communication, computing, and caching. However, due to the vehicle's constraints, i.e., low computational resources coupled with a limited onboard battery, executing computation-intensive and delay-sensitive tasks at powerful computing paradigms (i.e., CC, EC, and VC) may significantly ameliorate the vehicular applications' performance. This section gives an overview of CC, EC, and VC to realize such computation offloading.

三种计算范式：中心式云计算（高时延，低吞吐），边缘计算（靠近用户，低时延），车辆计算（一组车辆可构成计算资源池）

A. Centralized Cloud (CC)

The public cloud model has been widely successful, including Amazon Web Services (AWS), Google Cloud, Microsoft Azure, and Baidu [41]. These all have built sizable offerings for a wide range of applications, which are facilitating innovative, cost-effective, and scalable storage and compute solutions [42]. This cloud model enables enterprises to derive valuable insights from the data by using compute on a scalable, consumption basis. Cloud computing can improve computing performance by using abundant computing resources [43]. However, the cloud servers are far from the user that causes an unbearable delay, which further deteriorates the performance of vehicular applications, especially those having strict-latency and computation-intensive requirements [33], [44]. Moreover, if a large amount of data is transmitted, it will over-burden the limited network bandwidth.

Nowadays, vehicles are becoming more intelligent, and the urban infrastructure is gradually improved, enabling a large number of vehicular applications [45]. However, implementing these AI-based applications requires a lot of computation and storage capabilities, while the vehicles have limited capabilities in terms of computation and storage and cannot meet the growing resource demand. For instance, a vehicle sends data using the cellular network to the remote cloud for processing. However, due to network latency and reliability, such critical use cases require tens of milliseconds. In this situation, cloud dependence might deteriorate the application's operation and even undermine the safety application's efficiency if a minimum QoS cannot be guaranteed. For the aforementioned critical applications, edge processing seems to be a well-suited alternative to unlock several opportunities for a plethora of safety and non-safety vehicular applications.

依靠云计算，会增加时延，破坏安全应用程序的效率

B. Edge Cloud (EC)

The edge is a nebulous term, and the edge of the network is not a unique location; it can be a BS/RSU site, WiFi hot-spot, network router, nearby vehicles, IoT hub, or a local data center [46]. EC infrastructure can be leveraged to tackle the CC's salient issues, including processing delay, increased processing load, and information infringement. Edge computing in proximity to the users can play a crucial role in enabling guaranteed latency and throughput by

指在网络边缘，靠近用户的位置提供的计算、存储和分析服务，可以减少延迟，通过减少距离来提高服务质量，尤其是在处理延迟敏感和计算密集型任务时。边缘计算还提供上下文感知、支持移动性、实现高度响应的云服务、可扩展性、安全性和隐私性。边缘计算的基础设施可以是基站/路边单元（RSU）站点、WiFi热点、网络路由器、附近车辆、IoT中心或本地数据中心

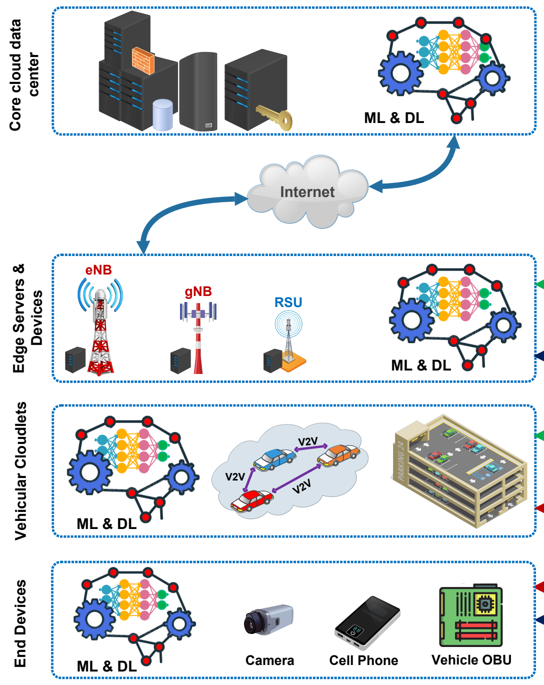

Fig. 2: Illustration of computing architectures. CC has powerful storage and processing power for AI-based analytics. EC-based MEC servers are in users' proximity to support URLLC applications, and VC is based on vehicles' computation resource pools.

reducing the distance [47]. Moreover, edge computing provides context-awareness, supports mobility, enables highly responsive cloud services, scalability, security, and privacy. Almost all devices with surplus computational resources close to the vehicle's location can play an edge computing role, depending on specific computing tasks. For vehicles, the Road Side Unit (RSU) can act as an edge server to collect, process, and store data efficiently. At the same time, the vehicle itself has specific communication, computing, and storage resources. When a large amount of computation is required, the vehicle can offload computation-intensive, and delay-sensitive tasks to the EC server. Compared with CC servers, EC can greatly reduce applications' response time and effectively alleviate the network burdens.

VEC refers to the merger of MEC with vehicular networks, as shown in Fig. 2. VEC is a promising paradigm in fulfilling the users' demands by ensuring low latency and high bandwidth [33]. In addition, VEC can figure out the conflict between limited vehicle resources and the increasing vehicles' computational demands [48]. However, compared with cloud computing, the computing power of edge nodes is limited. The emergence of EC processing must not be mistaken for the remote cloud's demise. Instead of delivering AI functionality, cloud computing will remain a vital entity in the foreseeable future. However, EC promising architecture opens up tremendous possibilities for AI-enabled applications, apart from improving the existing ones [49]. AI has also found its way to edge computing and changed it to intelligent edge enabling fast task processing including inference, pattern matching, decision making, and others. Integrating AI into edge can bring several advantages, i.e., prompt and real-time response on the edge-based AI, which can lead to handling higher QoE and confidence along with improved security. We further elaborate the edge cloud into three variants: MEC server, nearby vehicles, and hybrid MEC.

1) MEC Server: MEC servers are usually equipped with the eNB/gNB or RSU, core (i.e., 4G Evolved Packet Core (EPC)/5G Next-Generation Core (NGC), and gateway [33]. In the VEC network, the use of edge servers is the most common option for users to offload, which means installing RSU devices along the roads of the city to act as an EC server. Data transmission between the vehicle and RSU is performed via V2I mode. RSU based server has abundant computing and storage resources and can quickly and efficiently process requests from the vehicles without help from the remote cloud. In general, the service range of edge servers at RSUs is limited due to the vehicles' high mobility. For instance, in a scenario where a vehicle offloading task cannot be completed within the service scope of a given edge server, the vehicle has left the current RSU range before the task is completed. At this time, the MEC system needs to realize the perception of the vehicle's position, and the task needs to be migrated to another RSU where the vehicle is approaching [50].

2) Nearby Vehicles: In addition to offloading the computing tasks to the roadside MEC server at the RSU/BS, the vehicle can also offload it to the surrounding vehicles via V2V mode. We know that vehicles are equipped with computation and storage resources. However, in most cases, these resources are not effectively used. Therefore, if these idle resources can be fully utilized, the QoS of vehicle users can be greatly improved [51]. On the other hand, deploying a large number of RSUs will have high operating costs. Thus, leveraging surrounding vehicles as edge servers is a good choice. In reality, many users are reluctant to share their surplus resources with other users for various reasons. An effective reward mechanism ensuring security and privacy can enable others computing resources to be utilized.

3) Hybrid MEC (HMEC): HMEC leverages both offloading paradigms, i.e., tasks can be offloaded to the MEC server and the nearby vehicles [52] to get parallel computing. In this scenario, the vehicle can directly communicate with the RSU and the nearby vehicles through V2I and V2V modes, respectively. In this way, HMEC can also improve the utilization of scarce resources [52]. For instance, when the vehicle moves out of the RSU coverage, the task is offloaded to nearby vehicles, and if no vehicle is available around, then the vehicle will directly communicate with the BS using a vehicle to the Network (V2N) mode under LTE-V2X or New Radio (NR)-V2X communication.

车辆不仅可以将计算任务卸载到固定的MEC服务器，还可以根据实际情况，将任务卸载到附近的车辆上

TABLE II: Comparison of computing paradigms

Element	CC	EC	VC
Latency	Long	Short	Short
Computing capacity	High	Low	Low
Power consumption	High	Low	Low
Geo-distribution	Centralized	Distributed	Distributed
Access	Internet	Local network	Local network
Location-aware	No	Yes	Yes
Task multi-hop	Yes	Yes	Yes
Central Controller	Yes	No	Yes
Vehicle as Server	No	No	Yes

VC的核心思想是通过车辆网络（Internet of Vehicles, IoV）将一组车辆的资源形成一个云计算网络，实现资源共享

C. Vehicular Cloudlet (VC)

VC is different from the MEC server deployed at the RSU/BS to solve the problem of insufficient vehicle computing resources. A new idea based on vehicle network, vehicles' storage and processing units are leveraged to construct VC [53]. Vehicles' onboard computing is becoming more powerful for intelligent and connected vehicles. As shown in Fig. 2, VC computing consists of a shared pool of underutilized vehicles' resources that include computing, storing, sensing, and communicating devices [54], [55]. The resources of a group of vehicles can form into a cloud computing network through the Internet of Vehicles (IoV) to realize resource sharing. This vehicular cloud is very similar to traditional cloud computing with the difference that the computation resources are provided by vehicles. If VC is created on a large scale, then vehicles can self-organize the network, which can save many computing resources and make the vehicular network more intelligent [56].

Vehicles on the street may be dynamic or static, and based on that, VC can be defined accordingly as dynamic VC and static VC, respectively. For example, many vehicles are in some parking lots, and these vehicles can provide services by forming a VC. The parking lot has one or more access points (AP), which can send the status of the VC to the data management center in real-time. For static VC, one of the challenges is to predict the parking time of the vehicle to allocate computing resources reasonably [57], [58]. In dynamic VC, vehicles are moving on the road, which can be used as carriers/computing nodes for transmitting information and computing tasks, respectively. They can continuously transmit information by establishing new connections [59].

These resource-rich vehicles are resource providers, which can work as data centers and provide on-demand computing resources on-demand [60]. Similarly, for hosting vehicular applications, RSUs can be part of the VC computing and can also rent their resources [61]. However, for VC to be widely adopted and have a significant social impact, security and privacy issues need to be resolved [62]. Moreover, for persuading vehicles owners to lend their computing resources to the VC, some form of rewards should be given. For instance, if a new vehicular user enters the parking lot and agrees to lend resources may get free parking [63]. For readers' convenience, a comparison of these three kinds of cloud is shown in Table II.

需要解决安全和隐私问题。此外，为了鼓励车辆所有者共享他们的计算资源，可能需要提供某种形式的奖励

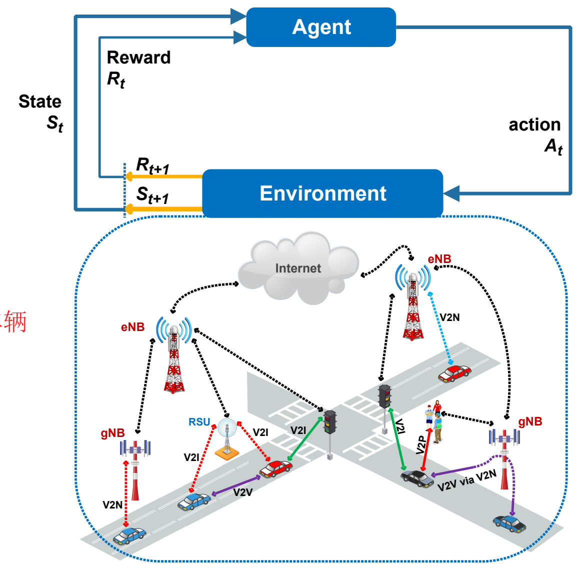

Fig. 3: RL depiction showing the state, action, and reward within the VANET scenario for task offloading environment.

III. Overview of RL/DRL Algorithms for Vehicular Task Offloading

RL is inspired by behavioral psychology and aims to find a balance between exploration and exploitation [64]. One of the challenges in vehicular task offloading is the dynamic environment [65], which is hard to guarantee the accuracy of estimated results with a traditional algorithm. However, the RL method can solve the problem of optimal resource allocation in a large and complex environment by learning directly from the environment. For example, through the collection and processing of real-time traffic data, the route can be intelligently planned for users by RL decision to avoid traffic congestion [51]. An overview of RL and DRL techniques will facilitate readers in understanding the fundamentals of AI technology for vehicular task offloading. In this section, we first introduce RL (III-A) and focus on the most common RL algorithms (III-B). These algorithms are the most widely used in vehicle task offloading environment. Then, we have a brief introduction to DL (III-C) and discuss DRL algorithms, including value-based (III-D) and policy-based (III-E) algorithms.

A. RL Basics

Markov Decision Process usually describes RL (MDP) [64], which is a process of stochastic control of state and action over discrete time. Generally speaking, MDP mainly includes four elements: state, action, transition probability and reward. For better illustration, Fig. 3 shows a Vehicular Adhoc Network (VANET) scenario considering task offloading environment state (i.e., the location of vehicles, the RSU resource, and the spectrum resources) in time-slot t expressed as s_t. The agent chooses an action a_t to interact with the offloading environment. At the next time-slot, the environment state changes to s_{t+1} according to the transition probability and returns a reward r_{t+1} to the agent. The agent modifies the next action according to the reward until it finds the optimal policy π, which is to maximize the long-term cumulative reward [66].

In MDP, $G_t = R_{t+1}+\gamma R_{t+2}+\cdots+\gamma^{T-t-1}R_T$ represents the sum of all the rewards on a Markov chain from time t onward, where γ is discount factor and $\gamma \in [0,1]$. In order to measure the value of each state, state value function $v(s) = E[G_t|S_t = s]$ is defined that represents the expectation of G_t from the current state to the end of the Markov chain. Then, the famous Bellman equation is derived as:

$$V(s) = R(s) + \gamma \sum_{s'\in S} P(s'|s)V(s'), \tag{1}$$

where S is the set of all the states at the next time. $P(s'|s)$ represents the transition probability from state s to next state s'. From the Bellman equation, the state value function $V(s)$ consists of two parts: 1) the reward $R(s)$ at the current state and 2) the expectation of the state value at the next moment.

In addition, $\pi(a|s) = P(A_t = a|S_t = s)$ represents the probability of a certain state s taking possible actions a. In particular, the policy includes all the actions and probabilities of the agent in each state. In the Markov chain, the policy is decided by the current state and has no relation with historical information, and the agent can change the strategy over time. Similarly, we define $q_\pi(s,a)$ as the action-value function, which is the expectation gains obtained by choosing a specific action a in state s by following policy π.

B. Classical RL Algorithms

Solving an RL problem means to look for an optimal policy so that the agent will always gain more than other policies in the process of interacting with the environment. It can be defined that the optimal state value function is the largest among many state value functions produced under different strategies:

$$v^*(s) = \max_{\pi} v_\pi(s), \tag{2}$$

where $v^*(s)$ represents optimal state value in state s.

1) Value Iteration (VI): How to choose the best policy, and the easiest way is to opt greedy method, where the agent always chooses the actions giving the most outstanding value. From (1), we can see that for finding the current state S; we first need to obtain the value of the following state S'. The problem has the characteristics of optimal substructure and overlapping sub-problem, so the algorithm based on dynamic programming can resolve the problem. The main idea is to use (1) and calculate the system state. Then, use the greedy method to select the maximum state and update the current state until finding an optimal policy π. The VI is expressed as follows:

$$V_{i+1}(s) \leftarrow \max_a \left(R(s,a) + \gamma \sum_{s'\in S} P_{(s'|s,a)}V_i(s')\right). \tag{3}$$

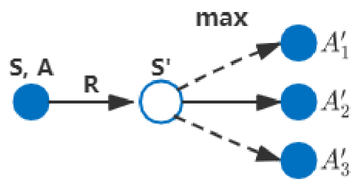

Fig. 4: Q-Learning's state, action, and reward depiction.

The VI belongs to the dynamic programming (DP), and the policy iteration method also belongs to DP. The DP-based method needs to be iterative constantly. Therefore, it is not efficient in environments with complex states, such as dynamic vehicle environments. Furthermore, the VI algorithm is model-based [67], which means we can get the environment's transition probability matrix P. In contrast, if there is no state transition probability, it is called model-free.

2) Q-Learning: There is no state transition probability P for model-free. In this situation, the agent needs to interact with the environment directly. After obtaining a large number of trajectory τ information, and can then estimate the real R and P, which are Monte Carlo (MC) and Temporal Difference (TD) interaction modes. The agent must collect complete trajectory information for MC before updating the state value. On the contrary, the TD method does not require complete trajectory information to estimate the state value online, which is fast and flexible. Moreover, TD iteration is also used for the classical Q-Learning algorithm. The TD iteration is given as follows:

$$v(S_t) \leftarrow v(S_t) + \alpha(R + \gamma v(S') - v(S_t)), \tag{4}$$

where $R+\gamma v(S')$ is called TD target and $R+\gamma v(S')-v(S_t)$ is called TD error. We can see that it replaces the value of the current state with the immediate reward and the estimated value of the next state.

For Q-Learning [68], the agent chooses an action A on the state S by $\epsilon - greedy$ method. Then the system change to the next state S', and the agent gets the reward R. Then, Q-Learning uses the $greedy$ method to update the value. Specifically, it uses the $greedy$ strategy to select A'. Choosing the action a makes $Q(S',a)$ have the most significant value. Corresponding to the Fig. 4 is to select one of the three blue circle actions that make $Q(S',a)$ the largest as A'. The iterative update formula is as follows:

$$Q(S,A) += \alpha[R + \gamma \max_{a'} Q(S',a') - Q(S,A)]. \tag{5}$$

It is important to note that the action A' selected now will not be executed in the future but is used to update the Q table. As can be seen from (5), there are two different strategies. A strategy π is to choose an action with the maximum q (like the $greedy$ method), which we call the target policy. Another behavior policy μ can interact with the environment to generate different trajectories with more exploratory properties (such as the $\epsilon - greedy$ method).

Off-policy uses two different policies (i.e., Q-Learning) to deal with the process of exploration and optimization in RL. The other one is on-policy, in which exploration and optimization use the same policy. However, since the target policy and behavior strategy are different and off-policy based algorithms can repeatedly use the past data. Therefore, in a real-time changing vehicle environment, Q-Learning can guarantee the global optimal solution to a greater extent.

C. Deep Learning (DL)

DL [69] is a subfield of ML concerned with algorithms inspired by the structure and function of the brain called artificial neural networks (NN), which consists of the input layer, one or more hidden layers, and output layer. A NN can be seen as a mathematical model that takes input data X and gives output result in $\hat{Y}$. The input data is subjected to matrix multiplication in each layer and called forward propagation. The learning process of the model is constantly adjusting the parameter θ in the network and produces a favored output $\hat{Y}$ that infinite proximity target data Y. The distance between $\hat{Y}$ and Y is usually quantified using a loss function: $L(\theta) = f(\hat{Y}(\theta), Y)$, where f represents different distance functions, i.e., Mean Square Error (MSE), Mean Absolute Error (MAE), Cross-Entropy, and KL-divergence loss. The objective of the NN is to minimize the loss function, which is based on the gradient descent algorithm. The derivative $\Delta_\theta L(\theta) = \frac{\partial L(\theta)}{\partial \theta}$ is calculated, which is usually used to update the parameters:

$$\theta \leftarrow \theta - \alpha \Delta_\theta L(\theta), \tag{6}$$

where α is a hyper-parameter called step size. Where $\Delta_\theta L(\theta)$ is the gradient of the network parameter θ, and it's the greatest direction of the gradient. At the beginning of the algorithm, a random parameter θ_0 is initialized, then follows (6) for constantly updating the parameters until smallest θ is attained.

Common NNs include Convolutional Neural Networks (CNN) [70] and Recurrent Neural Network (RNN). CNN processes image data and performs convolution and filtering operations on the input matrix data, reducing the number of network parameters and improving the model's performance. RNN is also a special kind of NN, which not only considers the input at the last moment but also has the memory function of the previous input data [69]. It is mainly used in the field of Natural Language Processing (NLP) [71].

D. DRL Value-Based

Q-learning is a model-free reinforcement learning that can efficiently learn the value of a particular state's action. The aforementioned traditional RL methods can only be applied to simple scenarios. When the interactive environment becomes complex and includes many states and action space, the Q-Learning algorithm needs to maintain a huge Q table to store the action-values pair for each state. Therefore, to obtain the optimal policy, the agent must often revisit each state-action pair, which is not feasible for large state spaces. Consequently, data and frequent search operations require a large amount of memory space and run time, making the algorithm inefficient. Another drawback of Q-Learning includes handling discrete states only and is not applicable for continuous states. An excellent way to tackle RL issues is to integrate the NN into the RL algorithm and replace the Q table:

$$\hat{Q}(s, a, w) \approx Q_\pi(s, a), \tag{7}$$

where w is the NN parameter and $\hat{Q}(s, a, w)$ is the NN's output, which is approximately equal to $Q_\pi(s, a)$. The algorithm can adapt to continuous environments since no specific Q values are stored. The key benefit of NN is the ability to compute the Q-values in a given state for all possible actions with a single forward propagation process rather than visit the state-action pair frequently in a big table, thus improving the algorithm efficiency.

1) Deep Q-Network (DQN): Mnih *et al.* [72] proposed the DQN algorithm in 2015 and achieved excellent results in the ATARI game. The authors used experience replay and fixed target to make the model converge easily. The input data is usually a sequence of highly correlated states [73] and makes agents have poor generalization ability. Through the experience replay mechanism, the agent stores the experience $< s, a, r, s' >$ and interacts with the environment at each time in a data set. Then take a batch of data randomly from set to train the model, which reduces the correlation between training data [74].

In DQN, the parameters are updated as:

$$w \leftarrow w + \alpha(Y^{DQN} - \hat{Q}(s, a, w))\nabla_w \hat{Q}(s, a, w), \tag{8}$$

where the TD target Y^{DQN} is $r + \gamma \max_a \hat{Q}(s', a, w)$ and it's as a label. The $\hat{Q}(s, a, w)$ is the network that needs to be optimized, which is called the estimation network. It can be seen that the target network and the estimated network use the same network and have the same network parameters w. When the estimated network parameters are updated, the VALUE of the TD target will also change so that the network is difficult to convergence. In order to make the training process of the model more stable, the fixed target network method is adopted in DQN. Specifically, the network $\hat{Q}(s, a, w)$ is the estimation network with parameters w and adds a target network $Q(s, a, w)$ with the parameters w^-. At the beginning of model training, different frequencies are used to update the two network parameters to fix the target network parameters within a certain time. The estimation network to predict the $\hat{Q}(s, a, w)$ and the target network are used to calculate the TD target Y^{DQN}. Then calculate the loss function and leverage gradient descent to update the estimation network parameters. After repeating several times, copy the estimation network parameters to the target network $w^- \leftarrow w$.

2) Double DQN (DDQN): Although the DQN algorithm has achieved good performance in Atari games [73], it still has shortcomings. There exist an overestimation problem in the calculation of the TD target. The reason is the

dqn缺点：在TD目标的计算中存在高估问题

$\max_a Q(s', a)$ operation in the Q-Learning algorithm, which causes the Q value to be larger than the real Q and results in overoptimistic value estimates. Because it is not always the best strategy to choose the action with the largest Q value before the network converges.

Hasselt *et al.* [75] proposed an improved algorithm for DQN known as DDQN. Improvement of DDQN is achieved by using two NNs for the selection and evaluation of the action with different value functions.

$$Y^{DDQN} = r + \gamma Q(s', \arg\max_a \hat{Q}(s', a, w), w'). \quad (9)$$

Here, we use an estimation network with parameters w to select the action a with the maximum Q value, then evaluate the selected action a in the target network with parameters w^-. Which makes it possible to reduce the overestimation of the Q value while increasing the DRL's model stability [76]. DDQN algorithm gets the benefit of double Q-Learning and keeps the rest of DQN algorithm [74].

E. DRL Policy-Based

Analogous to our process of approximating the value function using NN: $\hat{Q}(s, s, w) \approx Q_\pi(s, a)$. For the Policy-Based RL method, we can use the same method to approximate the policy, given a $\pi_\theta(s, a)$ approximated by the parameter θ, and then find the best policy θ. The agent collects many trajectories τ to interact with the environment by MC sampling first. Then, to maximize the total reward in each step. The policy gradient estimator obtained by MC sampling can be expressed as:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{T-1} G_t \nabla_\theta \log \pi_\theta(a_t|s_t)\right], \quad (10)$$

where $\nabla_\theta \log \pi_\theta(a_t|s_t)$ is called score function and G_t is the sum of rewards in each step which indicate the current score function is good or bad. We expect that in the optimization process, the strategy will enter as much as possible into the area that gets more rewards. A very classic algorithm called REINFORCE was proposed by Williams *et al.* [77] in 1992.

1) Actor-Critic (AC): The REINFORCE uses MC sampling and has a high variance. Another disadvantage is sampling utilization rate is not high. So one of the improved methods is to use the TD sampling method and Q value to replace the MC sampling and G_t in (10), respectively. In this way, the algorithm both has value approximate and policy gradient, and it is called AC algorithm, which is as follows:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\sum_{t=0}^{T-1} Q_w(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\right]. \quad (11)$$

Here (11) contains two parts, i.e., actor and critic, which are given as:

- Actor: The action network with parameter θ. Adjust the output based on the actor's feedback on the action.

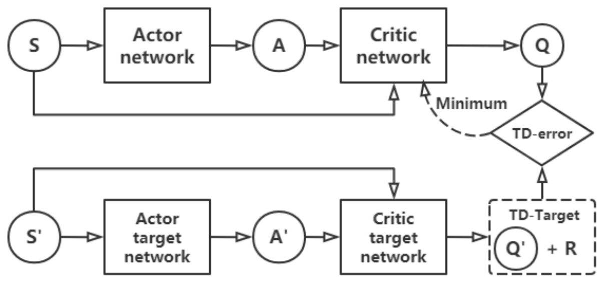

Fig. 5: DDPG algorithm structure: based on Actor-Critic architecture and using target network.

- Critic: The network with parameter w and function is to evaluate the action value of the actor-network output. It is similar to the role of a critic.

The variance caused by using TD sampling can be introduced into the baseline:

$$A_{\pi,\gamma}(s, a) = Q_{\pi,\gamma}(s, a) - V_{\pi,\gamma}(s), \quad (12)$$

where $A_{\pi,\gamma}(s, a)$ is call advantage function, which represents the advantage of the current action value over the state value. Therefore, the $Q_w(s_t, a_t)$ in (11) can be replaced by $A_{\pi,\gamma}(s, a)$ in (12) [78]. This is called Advantage Actor-Critic (A2C). The critic network does not need to estimate Q, but to estimate V, and the loss function for critic network can be expressed as $L^{Critic}(w) = (\delta)^2$. The parameters updated formula for critic network is like (8) in DQN.

The AC algorithm uses the TD method for sampling, which can use fewer samples to learn with less computational resources to select an action [74]. Moreover, it is a stochastic gradient algorithm, which can handle a continuous action environment. Improved algorithms based on the AC architecture include asynchronous advantage actor-critic (A3C) [79] and soft actor-critic (SAC) [80]. In addition, another way to improve REINFORCE is called Proximal Policy Optimization (PPO) algorithm; it uses an important sampling method to improve sampling utilization and have good performance.

2) Deep Deterministic Policy Gradient (DDPG): The random strategy is to sample the obtained strategy distribution. However, frequent action sampling is computationally intensive for some high-dimensional continuous-value actions. The DDPG [81] algorithm combines the advantages of both AC architecture and deterministic strategy, which can learn competitive policy by exploiting low dimensional observations. In the DDPG algorithm, use actor-network to approximate the operation $\max_a Q(s, a)$ in the DQN algorithm: $\max_a Q(s, a) \approx Q(s, \mu(s|\theta))$, where θ is the parameters of policy network. The critic network is utilized to estimate the Q value, which indicates how good the action output is from the actor-network. Like the DQN algorithm, DDPG has also experienced replay and the target network. In this way, DDPG contains four NNs: actor-network, actor target network, critic network, and critic target network.

TABLE III: Comparison of RL/DRL algorithms

Algorithms	Variants	Main Characteristics	Advantages	Disadvantages
VI	Policy Iteration(PI)	Use Dynamic programming method	Faster iteration efficiency	It relies on transfer matrix and is inefficient
Q-Learning	SARSA, Dyna-Q	Use Q table store all the Q value of action	Use TD sample and flexible and efficient sampling	It's inefficient in the continuous state
DQN	Duling DQN	Use NN instead of Q table	Fast convergence and easy to implement	Over estimation for Q values
DDQN	/	The evaluation and selection of actions are implemented by dual networks	Convergence process is stable and fast	Deterministic strategies do not satisfy all cases
REINFORCE	PPO	Stochastic Policy	learning optimal strategy directly	With a high variance and low sample utilization
AC	A2C, A3C	Both policy update and value update are included	Higher update efficiency can handle continuous action environments	It is not easy to convergence
DDPG	MADDPG	Process continuous actions with deterministic strategies	Easy convergence and stability	Deterministic strategies cannot explore the action state space well
VDN	QMix	The total Q value is the sum of all agent Q value	Speed up the task learning	The total Q value fitting ability for complex environment is limited
QMix	/	The total Q value is obtained by mixing the Q values of all agents	Using mixing network has better performance for total Q	Usage scenarios are limited

As shown in Fig. 5, the actor-network input is the environment state, and the output is an action. This action input into the critic network is leveraged to obtain a higher Q value. For the actor's loss function, the larger the Q value of the output action, the better the action and the smaller the loss. The loss function of the actor-network is defined as:

$$J^{Actor}(\theta) = -\frac{1}{m}\sum_{i=1}^{m} Q(s_i, a_i, w), \tag{13}$$

where m is the batch size, and θ are the parameters of the actor-network. The update of the critic network is similar to DQN (8). It is worth mentioning that the parameter copy from the current network to the target network in DDPG uses a soft update method. Specifically, the parameters are updated only slightly each time $w' \leftarrow \tau w + (1-\tau)w'$, where τ is the update parameter, which is generally smaller. Hence, target values update slowly and optimizes the learning stability [82].

F. Multi-Agent Reinforcement Learning (MARL)

MARL is a hot research area, which provides another perspective on systems [83]. Compared with a single agent, MARL can speed up task learning utilizing communication [84], teaching [85] and imitation [86] among multiple agents, and achieve better performance. In addition, the parallelization technology [87] can also accelerate the efficiency of the agent to explore the environment. MARL decision-making process can be expressed as a partially observable Markov decision process (POMDP); each agent intelligently obtains part of the observation of the environment, and all agents select actions according to the observation to form a joint action. Then use joint actions to interact with the environment to get an overall reward.

Multi-Agent DDPG (MADDPG) is improved based on the DDPG algorithm. The global state can train a single actor for the critic network. When testing, the actor takes local observations and follows the centralized training and distributed execution process. In addition, there are several MARL algorithms based on value decomposition that integrate each agent's value functions and get a full action-value function for all agents. The value-decomposition networks (VDN) [88] is the total action-value function is directly accumulated by the value function of each agent. The algorithm QMix [89] gets the total action value by a NN to mix the action value of each agent.

Although MARL has many advantages, a significant challenge is the curse dimensions caused by multiple agents. Specifically speaking, the joint action space increases exponentially with the number of agents and is generally more challenging to converge than a single agent. Besides, another problem is that it is difficult to define a learning goal [83]. Moreover, other agents' strategies can influence the optimal strategy of one agent [90], so agents must learn to cooperate to maximize the reward. The advantages and disadvantages of the above algorithms are compared in Table III.

IV. RL/DRL Solutions For Vehicular Task Offloading Using Network Edge Nodes

In general, offloading of workloads in mobile/vehicular networks means the transferring of tasks/computations to other mobiles/vehicles/servers for processing [4]. Vehicular task offloading means the sending of computation-intensive and data-intensive application components to a resource-rich computation infrastructure for execution [5], [6]. However, today's vehicles are well-equipped with different sensors, communication, computing, and storage modules. However, individual vehicles are still constrained to meet complex and innovative applications/services, such as augmented/virtual reality, voice/image/face recognition, or interactive games [91]. Among other vehicular critical applications, including autonomous driving, computing, and transmission delay, are

任务卸载的过程：将卸载的任务发送到资源丰富的设备/基础设施，在计算基础设施上执行任务，并将结果返回给源车辆。

the significant challenges to ensure the Quality of Service (QoS). Therefore, enhancing vehicular computing capabilities has become particularly important. In this situation, the task offloading technology can meet this need.

Specifically, task offloading includes the following procedures, i.e., sending the offloaded task to the resourceful devices/infrastructure, executing the task at the computing infrastructure, and getting the response in the form of the results back to the source vehicle. Considering vehicle available resources and the application's complexity, it can make any of the following three offloading decisions:

- Local execution, where the entire computation is performed locally at the vehicle. 本地执行
- Complete task offloading, here, the entire computation is offloading and processed by the edge nodes (i.e., MEC servers/vehicles). 完全由边缘计算节点完成
- Partial task offloading, some computing tasks are processed locally by vehicle, and the rest are offloaded to the other vehicles/MEC servers. 部分本地，部分服务器

According to the edge nodes' types, in this survey, we divide the state-of-the-art research into three categories: schemes leveraging MEC server (IV-A) for task offloading, using nearby vehicles (IV-B), or utilizing both MEC server and vehicle (IV-C) at the same time, which is called Hybrid MEC (HMEC). Based on each edge node type and the corresponding RL/DRL literature is then segregated into the value-based and policy-based sub-categories.

A. RL/DRL Solutions Leveraging MEC Servers

MEC server is one of the critical entities in 5G architecture, which enables both task and traffic offloading by deploying and expanding storage and processing resources close to the vehicles to support delay-sensitive and computation-intensive applications [32]. The MEC server is a common location for vehicle selection tasks to offloading because proximity and availability support mobility and context-awareness. Consequently, MEC servers facilitate highly responsive cloud services, scalability, security, and privacy [92]. As shown in Fig. 6, vehicles can use V2I mode to communicate with RSU, thus offloading the delay-sensitive tasks having real-time restrictions to reduce latency. For vehicles, in addition to deciding whether to offload (full/partial task or perform local execution), when and where (cloud/edge infrastructure), it is also necessary to decide on which edge node to offload the application. In addition, because the coverage of RSU is limited and the vehicle is moving at high speed, the time required for task calculation may exceed the residence time of the vehicle in the edge node. asks need to be migrated between different RSUs leveraging virtualization technology, which is an exciting solution [93]. The following subsections investigate the advanced solutions into value and policy-based schemes under this category.

1) Value-Based Task Offloading: With the acceleration of urbanization and the application of various emerging technologies, our cities have undergone tremendous changes.

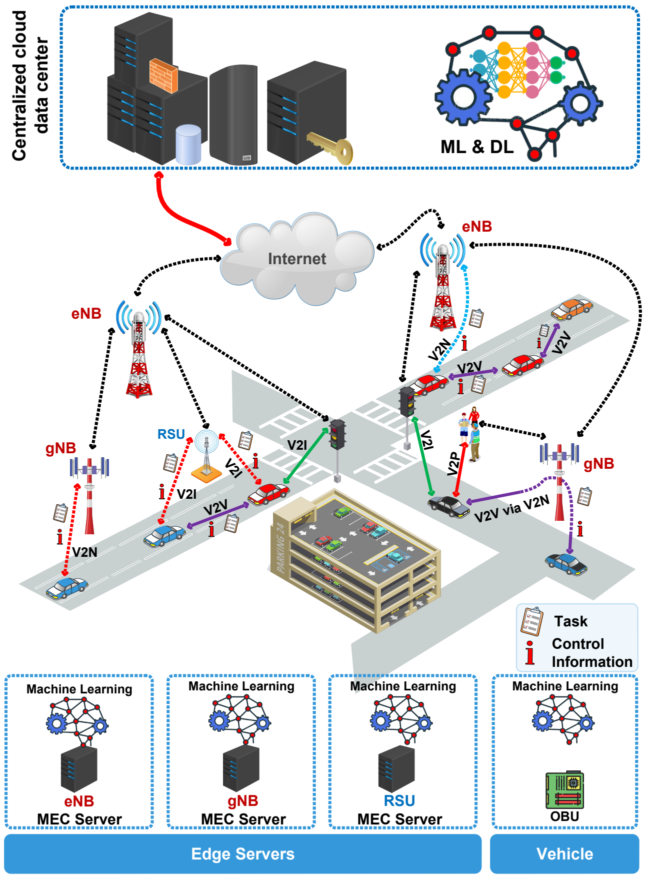

Fig. 6: Depiction of both EC and VC leveraging ML algorithms for vehicular task offloading scenarios.

By deploying the infrastructure on both sides of the road, a more intelligent transportation network can be constructed, the offloading of computing tasks can be realized, and the computing burden of vehicles can be reduced. However, the high mobility of vehicles makes multi-lane vehicles and resource allocation management among multi-lane vehicles a huge challenge. an *et al*. [94] modeled heterogeneous network based on mobility-aware coding probabilistic caching and computational offloading scheme. The authors aimed to minimize the system cost by jointly optimizing caching and computational allocation. Moreover, two constraints, including resource fluctuations and storage capacity limitations, are considered. Considering the complexity of the environmental action space caused by the mobility of vehicles, the authors proposed a twin-timescale framework, which used the Particle Swarm Optimization (PSO) technique and the DRL method to solve the problem in the large timescale and small timescale, respectively. The performance is validated, and the model showed better results than the random and even allocation techniques.

车辆的高机动性使得多车道车辆和多车道车辆之间的资源分配管理成为一个巨大的挑战

In [95], the authors modeled the traffic environment as a finite Markov chain, which was transformed into a joint optimization problem under constraints such as energy consumption and execution delay to maximize users' QoS. In addition, the authors used DDQN and dropout regularization to overcome the shortcomings of DQN that always overestimate the Q value. In their setup, the reward function equals the sum of the computation rate and the communication rate. The simulation results showed that the QoE metrics of the proposed scheme are about 20% and 12% percent higher than the greedy and DQN algorithms, respectively.

On the other hand, the work in [96] studied the limited cellular spectrum and energy supply to minimize the cost of offloading by reusing the cellular spectrum and reducing the traffic transmission. In order to ensure the user's QoS, tasks with urgent delay constraints are performed in the Macro-cell BS (MBS). After that, a distributed DRL algorithm is proposed to allocate network resources. The system status is the interference level of all the users. Moreover, all actions of data transmission over a channel selected by users are considered to maximize the long-term rewards and reduce communication between the vehicle and the MBS. According to the actual data of taxis in Hongkou District, Shanghai, this method indicated good performance in the simulation environment.

Dai *et al.* [97] studied the application of Multi-Armed Bandit (MAB) algorithms to resolve the task offloading in MEC. Considering the heterogeneity and the high mobility of vehicles, the authors proposed a distributed scheme. In this scheme, a table is created to record the task's estimated waiting delay before being processed by RSU, and then it is regularly updated via the feedback signal of the task assignment. Accordingly, the system knowledge-based table is used to derive a threshold-based probabilistic method to get the optimal task allocation. Comparing Local-First (LF) and Ordinal Sharing Learning (OSL) algorithms in traffic simulators proves that this algorithm has the best performance in various scenarios. Furthermore, the work can be extended by considering the packet loss and interference in communication.

On the contrary, Ibrar *et al.* [98] considered Software-Defined Vehicular based Fog (SDV-F) computing architecture and proposed an AI-based Resource and Task offloading NETwork (ARTNET) to allocate resource and optimize communication. As a result, ARTNET can provide reliable and low-delayed communication services in a dynamic environment. The framework is composed of agent and controller, support horizontal (fog-to-fog) and vertical (vehicle-to-fog or fog-to-cloud) offloading, minimizing communication delay while maximizing resource utilization. NN is then combined with the Q-Learning algorithm to achieve synchronization and task load balancing. The simulation results showed improved performance compared with the two baseline algorithms regarding task execution delay and energy consumption.

A different approach is put forward by Xiong *et al.* [99], assumed a heterogeneous vehicular framework and considered the offloading failure probability in the VEC network. The framework included three modes of communication network: DSRC, C-V2X, and millimeter-wave (mmWave). Moreover, random network calculations are used to derive the upper bounds of offloading delay under different technologies. Then, an intelligent offloading scheme is put forward to ensure reliability while reducing the resource cost. In addition, the authors proposed Sync Federated Q-Learning (Sync-FQL) algorithm to support partially training, which can accelerate the exploration of the motion space, optimize the model by learning different parts of the Q value, and avoid the long-term training process. This algorithm is based on traditional RL, which simultaneously calculates the state transition probability and maintains a Q table. From the simulation results, it is indicated that C-V2X technology has the best performance. Furthermore, the Sync-FQL performance vastly outperformed the baseline algorithms, i.e., Q-Learning and Asynchronous-FQL.

Khayyat *et al.* [100] proposed a scheme considering multilevel vehicular edge-cloud computing. In particular, the authors assumed one cloud server and some RSU based servers. Furthermore, computing tasks from vehicles can be performed locally or offloaded to the VEC or remote cloud servers. It can be seen that dealing with multi-users computation offloading will be challenging. Therefore, the authors take into account all the vehicles' trajectories. The optimization goal is the weight of communication delay and energy consumption. It is pointed out that the authors did not consider the situation when vehicles go outside the RSU range.

Differently from the works above, Ye *et al.* [101] investigated a two-timescale RAN slicing framework for computation offloading in the cloud-enabled autonomous vehicular network (C-AVN). For handling a maximum number of vehicles with high mobility, multi-tier networking architecture including MBS and Small-cell BSs (SBSs) are considered. The MBS server can provide intensive computation to all the vehicles on the road. The objective for small-timescale is to achieve computation load balancing, whereas the large-timescale is to maximize the total communication resource utilization in one planning window. In order to accurately represent the state of the environment, the task computing scheduling problem is modeled as an MDP problem with communication delay constraints. Next, a cooperative Multi-Agent Deep Q-Network (MA-DQN) algorithm is proposed to maximize long-term network-wide computation load balancing. Extensive simulation results are demonstrated under vehicles traffic load variations and indicate that their scheme's performance is better than the baseline approaches.

Finally, Yi *et al.* [102] designed a distributed semi-online computation model for the problem of how to cope with a vast number of data messages on resource-limited vehicles in real-time. Only one BS and multiple RSUs equipped with MEC servers are assumed in this framework. The vehicle communicates with the RSU through DSRC, and a vehicle can offload the task to BS/RSU or execute locally. They divided the task offloading problem into vehicle scheduling and computing resource optimization problems and proposed a Semi-Online task distribution offloading algorithm based on Dueling-DQN (SODO). This algorithm can work in continuous state space and predict vehicle behaviors, where the environment state includes resource, channel allocation rate, and task completion degree. The algorithm predicted different vehicle offloading behaviors and calculated the total reward value after the offloading operation. The simulation results indicated that the scheme could improve efficiency and reduce energy consumption compared to the greedy and random approaches.

2) Policy-Based Task Offloading: Ning *et al.* [103] focused on how to allocate computing resources in the face of real-time changes in a wireless network. A hierarchical edge computation and caching model is framed based on a single BS and multiple RSUs equipped with edge servers. Assuming that the BS has sufficient computing resources to cover all vehicles, both BS and RSU can simultaneously process user resource requests. Joint optimization of computing and caching is framed to find the optimal resource allocation policy and proposed a DDPG-based algorithm. Based on the real traffic data, the scheme maximized the profits of MNOs and ensured the user's QoS.

In [104], Ke *et al.* investigated how to maintain computational offloading efficiency in vehicular network. The authors proposed an adaptive computation offloading scheme based on the DRL model considering multiple random vehicles following the Poisson distribution. Vehicle services are covered by the RSU deployed at the roadsides, and the vehicle task can be partially computed at the onboard processor or sent to the MEC server. The authors tried to trade off the total cost of energy consumption, bandwidth allocation, and buffer delay to find the optimal policy. The local task execution and transmit power are considered for energy consumption computation. The DDPG algorithm applies the continuous action space for optimal policy. Moreover, the Ornstein-Uhlenbeck (OU) noise vector is applied to explore the environment better and improve exploration efficiency.

Different from the computational offloading efficiency work in [104], Zhan *et al.* [105] studied the task scheduling for high-speed vehicle scenario. A distributed sequential decision problem is framed by considering the communication between the vehicles and RSU. Specifically, the road is divided according to the scope of the RSU service. The authors framed the queue model for task vehicles. They introduced the state transition regarding data transmission and local execution. The problem is modeled as an MDP and used a PPO algorithm to learn through offline strategies using important sampling techniques. Through the optimization of long-term consumption, the scheme achieved better results. This paper can be extended to multiple vehicles scenario.

In resource-constrained vehicles, the execution of computation-intensive applications is still a challenging task. To tackle the problem, Zhu *et al.* [106] proposed a Multi-Agent DRL scheme for Computation Offloading (MDRCO) in vehicular edge network . They considered the uncertainty of the environment with multiple vehicles and reduced the overall task execution delay of the entire system. The state of the vehicle includes the vehicle's real-time location and mission information. The DRL algorithm is based on the AC architecture and equipped with a target network to improve the stability of the training process. The data center can centrally train the network of actors and critics, and then the vehicles execute task offloading decisions in a distributed manner. Their simulation results are based on real map data, which showed that their scheme reduced the latency by 21.7% and 10.2% compared with the nearest neighbor algorithm and the AC algorithm, respectively.

Unlike the work in [106] where multi-agent are involved, Li *et al.* [107] studied MEC based task offloading problem and proposed location-aware task offloading policy to ensure URLLC and computing services. The optimization goal is to cut down the decision space for scheduling. When a vehicle generates a computing task, it is sent to the nearest RSU for computation which can process it individually or jointly with other RSUs. The result of the computing task is returned to the vehicle, and during the entire process, the vehicle's status (e.g., speed, position) must be updated in real-time. The model states the amount of computed data, the vehicle's average speed, the delay incurred during task completion, the service delay, and the service failure penalty factors. A deep reinforcement learning DDPG algorithm combined with CNN is used to get a low complexity solution in continuous action space. The simulation results showed that the proposed scheme could achieve the lowest cost compared to the baseline approaches.

In another work, Peng *et al.* [108] studied the joint allocation of spectrum, computing, and storage in a MEC-based vehicular network. The authors considered the resource allocation problem from two different perspectives, i.e., Macro-eNodeB (MeNB) and edge node, both equipped with MEC servers. Moreover, owing to the vehicles' high mobility and the network status that changes over time, conventional optimization techniques cannot solve such complex situations. Therefore, the DRL algorithm and hierarchical learning model known as Hierarchical DDPG (HDDPG) is designed to solve this optimization problem. During the algorithm learning period, the resource allocation strategy is updated according to the feedback rewards of the obtained environment. Furthermore, the delay requirements and storage resources are met simultaneously for maximal tasks offloading and improved users' QoS. The simulation results showed that the proposed scheme could find the best strategy faster and converge the network. To simultaneously support multiple offloading tasks and ensure their QoS requirements, alternatively, Peng *et al.* [109] utilized Unmanned Aerial Vehicles (UAV) in the MEC network to assist in offloading tasks. Specifically, MEC servers' deployment at the MeNB and the UAV are considered to cooperate in making resource allocation decisions and computing resources to vehicles. The authors transformed the problem into a distributed optimization problem and considered the constantly changing state of vehicles and UAVs to enable agents to make decisions quickly. The authors proposed an RL-based

TABLE IV: Summary of RL/DRL Based Vehicular Task Offloading Papers Utilizing MEC Servers

Cate.	Ref.	Year	Offloading	RAT	Algorithm	RL/DRL Elements			Optimization Objective
						State	Action	Reward	
Value Based	[94]	2019	Binary	DSRC	DQN	The state of the available RSU/MEC server.	which RSU is assigned to the vehicle.	Cost of communication, storage and computation.	Minimize the system cost.
	[95]	2019	Binary	DSRC	DDQN	The communication channel gain and available computing capabilities.	The connection relationship among vehicles, RSUs and BS.	The lease cost of the spectrum bandwidth and the computing resource.	Maximize the comprehensive QoE of vehicles.
	[96]	2020	Binary	Cellular	DDQN	The interference degree of all users.	The select cellular channels for data transmission.	The long-term reward.	Minimize the offloading cost while satisfying the latency.
	[97]	2020	Binary	DSRC	Q-Learning	A table for each MEC that the estimated pending delay.	The available MEC servers to assigning the newly submitted task.	Task completion time.	Minimize task completion time.
	[98]	2020	Binary	DSRC	Q-Learning	/	/	/	Minimizing communication delay.
	[99]	2020	Partial	Cellular	Q-Learning	The assigned proportion of different offloading and the reserved bandwidth.	A traffic assignment of different offloading technologies.	The gain of selecting offloading technology.	Minimize the communication/computing budgets and the offloading failure probabilities.
	[100]	2020	Binary	DSRC	DQN	The computational task profile in each vehicle.	A vector about binary computational offloading.	The weighted sum of execute delay and energy consumption.	Minimize the sum cost of entire system.
	[101]	2021	Binary	DSRC	MA-DQN	The number of active vehicles, the set of uplink and offloading actions.	A vector about resource allocation decisions.	A negative function of the computation load balancing.	Maximize long-term network wide computation load balancing.
	[102]	2021	Binary	DSRC	Duling DQN	1.System resource allocation rate. 2.Channel allocation rate. 3.Task offloading completion degree.	A vector about offload decision for each vehicle.	The long-term cumulative discount reward value.	Minimize system energy consumption.
Policy Based	[103]	2020	Partial	DSRC	DDPG	The status of each vehicle and available resources of each MEC server.	The computational and caching resource, and bandwidth that MEC server allocates to vehicle.	Profit function include revenues, computation costs and the caching costs.	Maximize the benefit for the mobile network operator.
	[104]	2020	Partial	DSRC	DDPG	The buffer queue state in each equipment and the instantaneous channel state.	The transmitting and allocated power and proportion of bandwidth.	A cost function about energy consumption and buffer delay.	Minimize the total cost.
	[105]	2020	Binary	DSRC	PPO	1.local processing unit (LPU) 2.Data transmission unit (DTU). 3.MEC server state.	The task allocation of local execution (LE), remote execution (RE), and holding-on (HO)	The time and energy consumption of all the tasks.	Minimize the long-term cost in terms of a trade-off between task latency and energy consumption.
	[106]	2020	Partial	DSRC	DDPG	The vehicles real-time location and mission information.	A vehicle selection vector.	A functions about tasks information and actions.	Minimize the execution delay of the entire system.
	[107]	2020	Partial	DSRC	DDPG	1.The computing data amount in zones, 2.The average vehicle speed, 3.The delay for edge servers.	The index of receiver RSU, helper RSU, and deliver RSU.	The real-value cost.	Minimize the computing time.
	[108]	2020	Binary	DSRC	DDPG	The vehicle location and task information.	A vector about resource allocation decisions according to the policy.	A reward function about delay requirement and real delay.	Maximize the number of offloaded tasks with satisfied QoS.
	[109]	2021	Binary	DSRC	MADDPG	1. Vehicle and UAV's location . 2. Resource state of MeNB and UAV.	The resource allocation about UAV and MeNB.	A reward function about MeNB agent and UAV.	Maximize the number of offloaded tasks.

improved Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm, which can be trained offline to make decisions about resource allocation when executed online.

3) Summary: We investigated the available literature leveraging RL/DRL approaches for task offloading towards the MEC server, and their main features are summarized in Table IV. These features include the classification category, offloading type, radio access technology, AI algorithm type, RL/DRL elements (i.e., state, action, and reward), and the optimization objective. Based on the literature review, the proposed schemes in this category mainly focused on the reduction of service delay [97], [98], communication and system consumption costs [94], [96], [104], [105] as their optimization goal; and it is a requirement for computation-intensive and delay-sensitive real-time applications. However, in some works, including [95] and [108], the authors are concerned about the system's QoS to give users a better experience. Moreover, most of the schemes opted for binary offloading and considered DSRC mode, an efficient short-distance wireless communication technology.

B. RL/DRL Solutions Leveraging Nearby Vehicles

The embedded computing resources in vehicles are not fully utilized and can be exploited to help other vehicles. On the one hand, due to the high mobility of vehicles and the limited coverage of RSUs, tasks need to be migrated between different RSUs. This process will increase the energy consumption. On the other hand, in rural areas without RSU coverage, surrounding vehicles can be used for task offloading to improve resource utilization. Although the vehicle has a high speed relative to the fixed infrastructure, the connection time is short, but the speed difference between two vehicles is much smaller in the same direction along the the same road [110]. Therefore, such vehicles can have a longer connection time and can manage the computing of tasks distributively and cost-effectively [111]. Vehicles can compute task locally or offload to the nearby vehicles (depending on task intensiveness and resources availability) using V2V communication mode to save energy and accelerate task processing as shown in the Fig. 6. Now, we review the value-based task offloading techniques.

1) Value-Based Task Offloading: Tang *et al.* [112] considered the surrounding vehicles to tackle task offloading problem. The authors assumed that some vehicles on the road are task vehicles, and others are service vehicles equipped with sufficient resources and can provide computing services. Moreover, the tasks are assumed to have a linear, logical topology that must be executed sequentially. The cost of migration is formulated using probability theory, and a DQN algorithm is proposed to minimize the task delay and energy consumption. DQN uses experience replay technology, which is an offline algorithm that requires environment state, including the channel qualities, the task's size, and the sojourn time. For the state transition about sojourn time, Bayesian inference is used to deduce the sojourn time at the current moment based on historical data instead of considering the distance between the vehicles and vehicles' speed. The proposed scheme has largely outperformed the other three baseline schemes based on the performance results.

Different from [112], Shi *et al.* [113] investigated incentive-based task offloading problem leveraging nearby vehicles for computation. The authors framed a distributed sequential decision-making problem for task allocation among multiple vehicles via BS. For the proposed solution, DRL is opted that consider dynamic pricing for vehicles' motivation to share their computing resources, task and serving vehicles' cost, the V2V link-state, and any variation in the service vehicles' computing resources. The agent learns the states of the vehicles within the coverage area in real-time and estimates the wireless channels' states and the link duration between the task and serving vehicles to allocate tasks. The DRL framework considered a dynamic vehicular environment and maximized the task vehicle's cumulative reward by efficiently allocating the serving vehicles' computing resources. The proposed scheme's performance is validated via simulation under different configurations and compared with greedy and random algorithms.

In another work, Chen *et al.* [114] studied a distributed computation offloading scheme by utilizing the surrounding vehicles' resources. The surrounding vehicles' resources are regarded as the resource pool (RP), and these resources are scheduled under the First-Come-First-Service (FCFS) principle. The mobile vehicle can offload the task to the surrounding vehicle via V2V, significantly reducing the execution time. A complex task can be broken down into many small sub-tasks, converting these small tasks to get better execution time in RP to a min-max problem. Based on DDQN, a distributed offloading algorithm is designed to solve vehicles' task allocation problems. The simulation results showed that their proposed solutions had increased resource utility with less task execution time than the local computing and Partial Flooding Algorithm (PFA).

2) Summary: The literature on leveraging proximity vehicles for task offloading is investigated, and primary features are summarized in Table V. Under this category, cellular communication technology is used, which enables V2V communication leveraging LTE direct approach. Moreover, LTE-V2X and NR-V2X technologies can adapt well to the cellular communication between the vehicle networks and have good coverage, bandwidth, and stability. In [112], [113], [114], partial offloading is used, which means the task is split into multiple sub-tasks, which is executed locally or assigned to other vehicles and improve the offloading efficiency while reducing the response time. There are very few articles for this part using RL/DRL solutions because the use of nearby vehicles as task offloading targets exploits unused resources while reducing the burden on the MEC server. Therefore, in general, while tasks can be offloaded to nearby vehicles, tasks can also be offloaded to the MEC server in parallel, which is a hybrid MEC.

C. RL/DRL Solutions Leveraging HMEC (MEC + Vehicle)

In this category, the state-of-the-art literature leverages both edge nodes, i.e., MEC server and nearby vehicles, and is named Hybrid MEC. It is a cost-effective offloading paradigm by efficiently exploiting vehicles' extra resources and enhancing computing resource capability without any additional cost. RSU based MEC server is a fixed edge server deployed along the roadside, and vehicles can access it through V2I communication. In contrast, V2V communication is used to access the other vehicle's computing resources. In this way, even in places where RSUs are not deployed, vehicles can use nearby vehicles for task offloading, which improves computing efficiency. On the other hand, in areas where RSUs are deployed, the available computing resources will increase, and vehicles have more options for offloading. Below, we present the RL/DRL-based research works leveraging HMEC into two sub-categories: value-based and policy-based task offloading.

1) Value-Based Task Offloading: Tan *et al.* [115] studied issues of mutual communication, caching, and computational design to improve the cost and efficiency of vehicle operation. The authors proposed the mobility perceived reward estimation for the large-time scale model. According to the scenario, vehicles can offload tasks to the concerned RSU and nearby vehicles. If RSU cannot entertain, the vehicle can offload tasks to the BS servers. The system's state includes the number of RSU servers and vehicles and the available cache capacity. As an optimization problem, computational resources and deadline constraints are considered to minimize system consumption. Finally, a DRL algorithm is proposed based on DQN to solve this problem. Their proposed scheme's performance is validated and vastly outperformed the baseline approaches.

Different from [115], Zhang *et al.* [116] focused on the resource management and task offloading of multi-server edge computing. The authors designed a model that includes both cellular and LTE-V networks. In their model, vehicles can work on a different spectrum and enable multiple communication paradigms to transmit the task file to BS with V2B communication or turn to the RSU with joint V2V and V2R transmission. The total computation required for each MEC task queue is modeled to maximize the utility of the offloading system under a given delay constraint. The system state is a vector of the total computation required by the tasks queuing under each MEC, and the action is the set of task offloading strategies with various transmission modes and offloading targets. A redundant algorithm is designed to ensure offloading reliability. The performance results showed that their scheme could improve the system's utility.

TABLE V: Summary of RL/DRL Based Vehicular Task Offloading Papers Utilizing Nearby Vehicles

Cate.	Ref.	Year	Offloading	RAT	Algorithm	RL/DRL Elements			Optimization Objective
						State	Action	Reward	
Value Based	[112]	2020	Partial	Cellular	DQN	The computation task,the data size of task,and sojourn time.	An integer variable represents the chosen server.	A function about total cost.	Minimize the overall costs of delay and energy.
	[113]	2020	Partial	Cellular	DDQN	1.The signal noise ratio (SNR) of channel. 2. the remaining available time and computing resources.	The task allocation for each server and prices.	The mean utility of offloading tasks.	Maximize the cumulative reward of the task vehicle for offloading tasks.
	[114]	2020	Partial	Cellular	DQN	A vector about the total task computation in the queue of the vehicles.	The task offloading strategy for the task allocated to vehicles.	A function about compound task's execution time.	Minimize the time of task.

Liu *et al.* [117] believed that in the MEC environment, traditional servers are usually deployed at the roadside units, or cellular BSs have fixed locations, which may cause "service hole." In addition, there are many mobile vehicles in the city equipped with sufficient computing resources, which can serve as mobile edge servers and provide computing services for other users. Therefore, the problem is formulated as a semi-Markov process, where the states of the system are the available vehicles and edge server computing resources. Then, the DQN algorithm is used to realize the resource allocation strategy and maximize the VEC network utility. The proposed scheme performs better than pure vehicular edge servers and fixed edge server methods.

Unlike the works in [115], [116], and [117], Zhao *et al.* [118] considered a new type of contract incentive mechanism by combining resource contribution and utilization together. Those vehicles with higher cumulative rewards can get priority while offloading tasks, thereby avoiding task offloading conflicts caused by simultaneous decision-making. At the same time, due to the system's complexity, DQN based distributed scheme is designed to reduce the conflicts in the allocation of computing resources. The overall system's consumption is minimized by selecting the target vehicle to offload the task. The simulation results proved that this scheme could motivate vehicles to contribute computing resources effectively. Moreover, the action space becomes larger with the rise in the vehicles' number, which results in DQN's large output space and seriously affects the offloading strategy.

In [119], Luo *et al.* focused on maximizing the resource utilization in the VEC network by jointly considering communication and computing resources for data scheduling. For the VEC network scenario, RSUs deployment is assumed to provide computing resources for vehicles, and at the same time, vehicles can also perform collaborative computation with nearby vehicles. Based on this, a unified framework for communication, computing, caching, and collaboration is proposed by considering the remaining life of data and the cache status. For data caching, the authors established a multi-queue model on the vehicle and RSU. The integration of communication resources and caching status reflects the relationship between data and cost. In order to get an optimal strategy, an RL-based DQN algorithm is utilized to minimize the system's processing cost. Through real-world vehicle data testing, simulations show that the scheme can significantly reduce the data processing cost. With the fast emergence of the IoT, vehicular networks have also evolved and become more advanced, but delay-sensitive and computation-intensive applications are usually not satisfied. Zhang *et al.* [120] observed that the existing offloading schemes lack in considering the highly dynamic vehicular network. Moreover, the different subjective wishes of vehicle drivers lead to different vehicle movement modes and speeds. Therefore, the authors proposed a synchronous random walk model to solve the task offloading problem, where the vehicles' position and speed follow uniform Poisson spatial distribution and Gaussian distribution, respectively. For minimizing all the tasks' execution delays, the DQN algorithm is presented. The results verified that the proposed approach could reduce the delay and perform well.

In another work, Zhang *et al.* [121] believed that the performance of existing task offloading schemes could not meet the vehicles' computing requirements. In order to improve the efficiency of computing offloading, the authors designed a car networking system architecture involving the factors of computing, communication, interference, and privacy to ensure the rationality of the model. In addition to sending offloading requests to RSU, vehicles can also send small computing tasks to nearby vehicles through V2V to improve resource utilization. In the face of constantly changing vehicular network conditions, the DDQN algorithm is designed to learn allocation strategies. Experiment results showed that this model can have the highest task offload rate and is more suitable for Internet scenarios.

Maleki *et al.* [122] proposed an online algorithm based on Q-Learning to solve the problem of vehicles, which cannot meet the necessary computation for some applications in the MEC environment. For minimizing the average processing delay, the authors formulated the problem as an MDP. In their model, vehicles can get periodic messages from nearby users, including vehicle speed, current location, and CPU frequency via DSRC mode. Furthermore, the authors assumed the block-fading Rayleigh channel model between vehicles and RSU. Finally, the total offloading time equals the sum of transmission delay (i.e., uplink and downlink) and execute delay. Simulation results revealed that the scheme has a faster convergence than upper-confidence-bound and adaptive upper-confidence-bound approaches.

2) Policy-Based Task Offloading: Ren *et al.* [123] introduced a Software-Defined Vehicle (SDV) Network architecture, which makes full use of the characteristics of SDN technology and can obtain VANET's global perspective effectively and dynamically. Moreover, SDN based network enables better management of network resources. The au-

TABLE VI: Summary of RL/DRL Based Vehicular Task Offloading Papers Utilizing HMEC

Cate.	Ref.	Year	Offloading	RAT	Algorithm	RL/DRL Elements			Optimization Objective
						State	Action	Reward	
Value Based	[115]	2018	Binary	DSRC	DQN	1) The states of the available RSU/MEC servers. 2) The available vehicles. 3) the available caches	A vector about allocation for each server.	The cost of communicationstorage and computation.	Minimize the system cost.
	[116]	2019	Binary	Cellular	DQN	A vector about the total computation required for each MEC.	The sets of task offloading strategies and offloading targets.	The utility of the offloading system.	Maximize offloading utility under given delay constraints.
	[117]	2019	Binary	Cellular	DQN	1) The available vehicular edge server. 2) The fixed edge server.	Computation task offloading strategy for selecting local computing	The VEC network utility.	The long-term utility.
	[118]	2019	Binary	Cellular	DQN	1) The vehicle speed, position and computing capacity. 2) Cumulative reward value.	A vector include the amount of computation resources contributed.	The cost of system.	Maximize resource contribution and resource utilization.
	[119]	2020	Partial	Cellular	DQN	The caching and position state of vehicles and RSU.	The scheduling actions for vehicular and RSU.	A function about total cost.	Minimize the system-wide data processing cost.
	[120]	2020	Partial	DSRC	DQN	The total processing delay of all the computation tasks.	The set of vehicles' offloading and communications decisions.	A function about task execute time.	Minimize the processing delay of computation tasks.
	[121]	2020	Partial	Cellular	DDQN	The computing power that each MEC server.	The offloading decision and resource allocation.	The long-term cumulative discount reward value.	Minimize the user cost.
	[122]	2021	Binary	DSRC	Q-Learning	The locations, speed and moving direction of vehicles and CPU frequencies for edge node.	A index represent the offloading decision.	Is inversely proportional to the minimum execution delay for the current state.	Minimize the average processing delay.
Policy Based	[123]	2020	Binary	Cellular	DDPG	The vehicle number, computing node resources and maximum service delay.	Decisions include offloading, service migration and resource allocation.	A function is negatively related to the execution time.	Minimize task execution time.
	[124]	2020	Partial	Cellular	DDPG	The energy consumption via V2V and R2V mode.	The decision-making about the task-data chunk.	Value of QoE.	Minimize energy consumption.

thors considered DDPG and framed an intelligent service offloading decision model. The model states include vehicle number, computing node resources, and maximum service delay. Also, action space includes service offload decisions, service migration decisions, and resource allocation decisions. The reward/goal of the system is to minimize the task execution time, which is the joint optimization problem of multi-cooperative service offloading. The DDPG based service offloading model simulation results have better performance and stability than the baseline approaches.

Faced with different edge servers and vehicle states, He *et al.* [124] proposed a novel QoE model restricted by energy consumption. The model considered three factors: cache space, computing power, and channel conditions. Specifically, surrounding vehicles can contribute computing resources as offloading carriers, but the unpredictability of vehicles brings excellent challenges to offloading services. Then, a DRL algorithm named PS-DDPG is proposed, using Priority Experience Replay (PER) and Stochastic Weight Averaging (SWA) mechanisms. For the PER mechanism, as the importance of the experience is different in the buffer, essential experiences should be prioritized to train the model for better resource utilization. Experimental results showed that their proposed algorithm has better stability and convergence and improves the QoE.

3) Summary: We have summarized the essential features of the research contributions after investigating HMEC offloading, as shown in Table VI. It is evident that the use of C-V2X wireless radio access technology for offloading is dominant because of supporting innovative applications and services [125]. LTE-V2X is part of 3GPP Rel. 14 and 15, while 3GPP Rel. 16 provided the new radio (NR) to support cutting-edge V2X applications, i.e., from the basic safety applications to the more advanced use cases posing strict and varying requirements of bandwidth, latency, and reliability [126], [7]. Moreover, NR V2X is supporting GHz bandwidth using mmWave for short-range communication, which can support the cutting edge demanding applications requiring extremely low latency, ultra-high reliability, and high throughput [18], [127]. From the table, it is pertinent to mention that most research works come under the value-based category using DQN algorithms specifically for binary offloading. In [115] and [119], considering the constraints like dynamic and limited storage capacities of computation resources, the authors focused on the system cost. While in [124], energy consumption is investigated during offloading, and in [121], the user cost function is to minimized. Similarly, in another work [116], the authors focused on optimizing system utility to ensure system reliability. Considering the importance of load balancing and server selection, which are the main issues in VEC systems, the authors in [120], and [123] focused on these issues under low latency constraints. Also, the gradient algorithm DDPG is used in [124] [123], thanks to the training stability brought by the DDPG algorithm using the target network.

V. RL/DRL SOLUTIONS FOR VEHICULAR TASK OFFLOADING USING VEHICLE CLOUDLET

Unlike using edge nodes for task offloading, vehicles with powerful computing capabilities can be used to form a cloud, which is called vehicle cloudlet [63]. In order to quantify the computing resources in VC, the computing resources provided by vehicles are measured by the number of resource units (RUs). The vehicular cloudlet paradigm can be beneficial for MEC servers in overloading situations, enhancing computing resources, and lowering the deployment cost. These resources come from the resource pool and are controlled and allocated by a centralized VC system. However, VC operates in a highly dynamic environment coupled with short and unpredictable sojourn time. For instance, a new resource unit will be available with the entry of a vehicle; when the vehicle leaves the cloudlet, the resources are unavailable. Consequently, the VC data center resource manager needs to migrate the tasks. As shown in Fig. 7, all vehicles form a computing network. These vehicles join the VC network and agree to lend their computing resources. In VC, dynamic resource allocation is quite challenging because resources are not provided in physical form (such as CPU and HDD) but are provided by time and usage after virtualization. In the following, we review the value and policy-based vehicular task offloading techniques are leveraging VC computing.

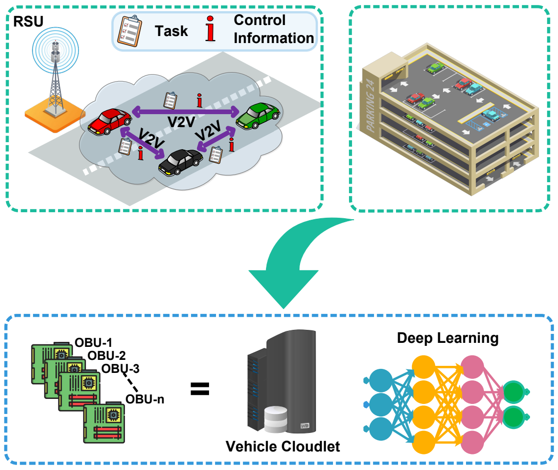

Fig. 7: RL/DRL based vehicular task computation scenario using vehicular cloudlet (i.e., static or dynamic).

A. Value-Based Task Offloading

Owing to the mobile vehicles' uncertainty, how to deal with the situation when the vehicle does not complete the task and leaves the VCC coverage area, Jiang *et al.* [128] proposed the task replication technique to handle the problem. The authors considered the deadline violation probability of the task as the optimization goal. Specifically, the task is collected by RSU and sent to vehicles based on policy. Poisson distribution is assumed for vehicles' arrival, and an independent exponential distribution is followed for the time remaining before vehicles meet the RSU. Moreover, the VI algorithm, a finite-horizon MDP-based solution, is leveraged to find the best policy. In addition, the Balanced-Task-Assignment (BETA) policy is proposed and proved that it is an optimal policy that always allocates tasks with the smallest number of task copies. The probability of deadline violation approximately obeys the Rayleigh distribution. However, the authors considered the same type of tasks and sizes. This work can be extended by considering realistic assumptions, i.e., different task types and sizes.

Sun *et al.* [129] studied the task migration in VCC network, where vehicles migrate the unfinished tasks to another vehicle before leaving the network. Since the prior knowledge of vehicular mobility is not available, the migration decision is made according to the vehicle location and computing capabilities. In addition, the computing task is described as a linear topology of sequential execution. Considering the uncertainty of vehicle motion and the heterogeneity of computing power, they developed a DQN-based algorithm to minimize the overall response time. The system status includes a task for migration, the vehicle that completed the previous task, and the location of all vehicles in the network. The action is an index representing the vehicle to whom the next task be migrated. The performance results showed a minimized overall response time. However, this paper only considered the migration process within VCC and did not mention the task collecting process. Different from the works in [128] and [129], public vehicles (such as buses) are considered as fog computing servers. Based on that Wang *et al.* [130] proposed a Vehicular Fog Computing System (VFCS). Specifically, public buses follow a fixed route and time point, which avoids the perception of vehicles and makes task offloading easier to achieve. At the same time, the system uses M/M/C priority queuing mechanism for resource volatility and application-aware delay requirements. A computing scheduler in the center of vehicular fog can decide whether to execute a task locally or send it to the remote cloud. Offloading environment is modeled as an SMDP. The status includes the number of fog computing servers and the number of fog servers allocated to the tasks with priority in the queuing system. Finally, the VI algorithm is used to obtain the largest long-term reward strategy. Simulation results showed that this method could significantly improve task offloading performance compared with other baseline methods. However, there are great challenges for traditional RL methods when dealing with complex problems.

Considering the vehicles and RSUs' heterogeneous computing capabilities, Lin *et al.* [131] studied the resource allocation of vehicle resources in the VCC system. Multiple RSUs and heterogeneous vehicles are considered in the model, and both of them can provide service. They quantified the computing resources in the system through the number of resource units (RUs). The service is executed by some RUs allocated by VCC or transferred to the remote cloud by V2I communication. In addition, they assumed that both the number of arrivals and departures of vehicles of each type follow Poisson distributions. The offloading process is based on the SMDP model. The objective function maximizes the long-term reward, the difference between revenue and the expected cost. The state includes arrival service requests, number of vehicles (i.e., different types), and some events. Since the VI algorithm is used to solve this problem rather than the DRL-based algorithm, they need to compute the transition probabilities before computing the state value. A new direction for extending this research is to check the impact of resource allocation's heterogeneity.

Ning *et al.* [132] leveraged parked vehicles' computation resources and proposed a three-layer architecture-based offloading framework, which focused on the entertainment services and minimized the total energy consumption while meeting users' delay constraints. The authors considered that RSU could collect the tasks and offload them to the remote cloud or parking and moving vehicles. The vehicle's arrival and the task flow both follow the Poisson process. The authors formulated the energy consumption model for the cloud, parked, and moving vehicles based on M/M/C queue theory. Due to the main problem's complexity, two sub-problems are defined. The first sub-problem is the flow redirection before making the offloading decision, which is solved by the Edmonds-Karp algorithm. The second sub-problem is to make offloading decisions by a sequence-based DRL algorithm. Finally, the model's validity is verified based on Shanghai's actual taxi trajectory data.

Unlike [132] flow redirection operations before make offloading decision, Liang *et al.* [133] proposed the secondary resource allocation mechanism. Specifically, the upper-layer architecture is SMDP-based, and the controller makes the decisions on a large time scale whether to receive the arrival service requests. The lower-layer architecture is the MDP-based spectrum resource redistribution mechanism, and it needs the system and channel state to make spectrum allocation decisions in each time slot. In the lower layer, the virtual unit represents the computing resource. Moreover, the arrival service request follows a Poisson distribution. The two layers can interact through reward feedback by their respective execution strategies. AI algorithms can adapt dynamic resource allocation through self-learning for optimization problems. The model finally uses the Dyna-Q algorithm to find the optimal allocation by maximizing resource utilization. The simulation results validated that the resource reservation strategy can improve QoE, and the secondary allocation mechanism can improve system performance. This model has a better allocation strategy than the greedy algorithm.

B. Policy-Based Task Offloading

Qi *et al.* [134] studied the problem of offloading using resource scheduling under constraints. Using DRL, the authors proposed a Knowledge-Driven (KD) service offloading decision-making framework. In the proposed scheme, the algorithm can directly learn online knowledge from the environment while considering the future data dependence of the current task and maximizing the long-term rewards. Heterogeneous resource nodes are considered that include BS, RSU, and vehicles with specific functions and parameters. The scheme included the hand-off times and a mobility model for vehicular cloudlets. At the same time, based on the A3C algorithm, the framework can simultaneously train multiple edge computing nodes in different places and then transfer the learned knowledge to the cloudlet's vehicles, which can better adapt to environmental changes. Numerical simulations proved that the KD algorithm largely outperformed the greedy algorithm with more dependencies between tasks.

To achieve effective resource allocation in VFC and meet the low-latency requirements of vehicle applications, Lee *et al.* [135] formulated the resource allocation problem as an optimization problem and proposed a task allocation model using the A3C algorithm. Specifically, facing the complex high-dimensional action environment, the high-dimensional continuous action space is converted into three-dimensional table data, which effectively improves the DNN's training efficiency. At the same time, due to the geographical distribution of the vehicular network, the use of the A3C algorithm can asynchronously train data in multiple locations, reduce the transmission of data in the vehicular network, and improve the learning efficiency. The final optimization goal is to improve user satisfaction, and the simulation results indicated that user satisfaction improved around 31% compared to the standard A3C algorithm.

In another work, Lee *et al.* [136] tackled the problem of service delays by using parked vehicles to allocate the computing resources. The authors used the RL technique and leveraged heuristic functions to improve the QoS in VFC. The vehicles can execute tasks by utilizing their local computational resources or sending them to BS or VFC. In their model, RSU can manage the mobility and computation resources and send resource update messages to the BS. The authors formulated an objective function for service satisfaction with a priority M/M/1 queuing system on each vehicular fog. Due to the high-dimensional information and continuous action space in the environment, RNN extracts resource availability patterns based on time and location. Moreover, RNN is integrated with the PPO algorithm. In addition, heuristic methods are used to accelerate the model training. In order to verify the algorithm's feasibility, actual vehicular data is used. The results showed that their proposed scheme is superior to the other traditional resource allocation schemes regarding service satisfaction.

C. Summary

From the investigation of the vehicular offloading literature using VC to compute tasks, we summarize the main features in Table VII. It is pertinent to mention here that maximum schemes, under this category, used value-based solutions [128]–[136] except the three schemes [134]–[136], which opted policy-based solutions. One of the key challenges in the VCC system is the mobility of vehicles. Therefore, in [128], the task replication technique is used to overcome the problem that the service process exhibits large uncertainty due to vehicle mobility. Different from conventional ways, in [130], the authors considered the buses as the fog nodes and provided computing resources. In [131], multi-RSUs and heterogeneous vehicles are considered in the system and derived the transition probabilities. While in [129] the task migration problem is focused when vehicles

TABLE VII: Summary of RL/DRL Based Vehicular Task Offloading Papers Utilizing Vehicle Cloudlet

Cate.	**Ref.**	**Year**	**Offloading**	**RAT**	**Algorithm**	**RL/DRL Elements**			**Optimization Objective**
						State	**Action**	**Reward**	
Value Based	[128] 2018	2018	Binary	DSRC	VI	1.The number of elapsed time. 2.The number of current replicas of task which are being processed.	A vector about decision for server.	The number of finished tasks in a time slot.	Minimize the deadline violation probability.
	[129]	2018	Binary	Cellular	DQN	The task and the current locations of all the vehicles.	A vector that decide which the next task should be migrated.	The overall response time	Minimize the overall response time.
	[130]	2018	Binary	DSRC	VI	The number of the fog servers at the current state and fog servers allocated to the tasks in the queuing.	A number represent the decision for task.	The difference between the revenue and system cost.	Maximize the long term expected reward.
	[131]	2018	Binary	DSRC	VI	Including service requests,number of vehicles of different types and event.	The number of RU allocated for arrival requests.	The difference of the revenue and expected cost.	Maximize the long-term reward.
	[132]	2019	Partial	DSRC	DDQN	1.The arrival rate of vehicles and tasks flows.2.The number of parked vehicles in range of RSU.	A number about offloading decision for task.	A function inversely proportional to energy consumption.	To minimize the overall energy consumption.
	[133]	2021	Partial	Cellular	Dyna-Q	1.The request event. 2. The number of virtual unit (Vus). 3.The number of important/ordinary service requests allocated to VU.	The model's response to and redistribution of received service requests.	The difference between system revenue and consumption.	Improve resource utilization and system QoS.
Policy Based	[134]	2019	Partial	Cellular	A3C	1.A tensor of the task.2.A tensor about all node.3.The moving speed of the vehicle.	A vector representing the execution position of a task.	The task execution delay.	Minimize the task execution delay.
	[135]	2019	Binary	DSRC	A3C	/	/	/	The user satisfaction for service latency.
	[136]	2020	Binary	DSRC	PPO	The vehicle movement and parking status.	A vector of the fraction of the computation load for service that is assigned to VF.	The service satisfaction obtained from a time slot.	Minimize the service latency.

may not be sufficient to accomplish the task. Unlike the common trend, in [132], the authors considered both the parked and moving vehicles as the server nodes in VCC and formulated the model based on M/M/C queue system. In [133], two-tier resource allocation architecture is considered for IoT communications and edge computing resources and implement service requests and spectrum allocation issues to guarantee system QoS, respectively. For improving training efficiency, [135] proposed a training method that decomposes the high-dimensional continuous action space into a three-dimensional grid. In [136], from the end users' perspective, the service delay is improved.

VI. Lesson Learned, Open Issues, and Research Directions

One of the main aims of our article is to understand how RL/DRL algorithms can be utilized for vehicular task offloading to improve the performance of vehicular applications and services. More specifically, different from the existing surveys in the underlying field, we endeavor to discuss the RL/DRL based vehicular task offloading and provide lessons learned along with crucial issues and challenges in the system design.

A. Lessons Learned

- In general, finding an optimal policy in a higher dimensional state space use by RL-based algorithms (i.e., DQN, DDQN) are better than the greedy and baseline algorithms [94], [100]–[102]. If the model supports partial task offloading, the policy-based algorithm (i.e., DDPG) is a better choice in an environment with highly dimensional action space [103], [104], [106], [107].
- Considering the complexity of the MEC environment, a good solution is to decompose the problem into two sub-problems: schedule offloading requests and resource allocation. The RL-based algorithm focuses on the latter problem to find the best policy. For the former problem, a two-sided matching scheme [95], [96] and flow redirection mechanisms [132] can be used.
- The optimization objective of the model is mainly execution time delay [96], [97] , [105]–[107] followed by overall system consumption [115], [119], [124]. Since most applications are time-sensitive in a vehicle environment, optimizing time delays is important for vehicle services
- For nearby vehicles [112], [114] and vehicle cloudlet [128], [129] [134]–[136] scenarios, the articles optimizing time delay accounted for the highest proportion. This is because both the service and mission vehicles are on the move, and most of the vehicles have less computing power, making the task more time-constrained.
- Asynchronous online learning can be performed by vehicles [134] and have less time delay. The new models can be updated at the edge computing nodes to better adapt to the environment. Although asynchronous aggregation algorithm has proven to be faster than the synchronous algorithm, which often converges to sub-optimal results [99].
- Vehicles' mobility prediction using AI tools (i.e., Bayesian inference and LSTM) [112], [136] can be employed to provide resources proactively or pre-fetch computation tasks and reduce the offloading delay.
- In the nearby vehicle, HMEC, and VC computing scenarios, multi-hop-based offloading is better in exploiting the computing resources than the single-hop transmission. Single-hop is more reliable but has more time delay after executing many times. So, there is

an intriguing trade-off in wireless hops and must be exploited according to the application requirements.

- The multi-timescale scheme is another good choice for the complex problems as it requires low system resources than the single-timescale framework [94], [115], [133].
- Unlike the MEC mode, in VC schemes, M/M/C priority queue is preferred for framing the task offloading model to enable better decision making. Poisson distribution is used for vehicle arrival rate, and the task service time follows the exponential distribution [130]–[132].
- There must be a technological orchestration architecture involving heterogeneous technologies, i.e., DSRC, cellular, and mmWave connectivity, to tackle the URLLC requirements posed by the heterogeneous nature of vehicular applications and services.
- Frequent tasks migration between vehicles can also add to the total cost [112] because of resource consumption. Multiple path selection algorithms should be adopted to find the optimal path for data sending and receiving the computational result at the source vehicle to minimize the number of task migrations.
- To provide an accurate evaluation of the proposed schemes, large-scale and realistic evaluation scenarios using practical data sets must be developed [132].

B. Open Issues and Research Directions

Despite remarkable advances in vehicular task offloading design using RL/DRL tools, many open issues are still to be tackled. These challenges are given as follows:

1) Modeling and Optimization: In general, an objective function needs to be subject to certain constraints; however, AI with Stochastic Gradient Descent (SGD) algorithm is not effective when the search environment is limited [49]. Modeling a task offload environment with real-time changes in network topology is considerably challenging. To approximate a complex environment to a mathematical model requires grasping the main variables in the environment is quite challenging. Some heuristic optimization algorithms (i.e., genetic algorithms) are usually deployed, but the convergence speed needs improvement. In the DRL algorithm, one of the problems is how to set up a reward mechanism. The agent needs to maximize the reward to make a decision, and what kind of reward mechanism can make a fair decision. How to determine the optimization goal of the model, optimize task execution delay or resource consumption, or perform joint optimization. Similarly, how to propose a suitable reward mechanism for different application scenarios is a problem that needs to be studied. Using appropriate reward functions and artificial intelligence technology can strike a balance between fairness and optimization [137].

2) Convergence of DRL: There are some problems with the DRL approach itself e,g., it is complicated to converge the DRL model for several reasons: 1) For approximating Q forms, NN is introduced in DRL, i.e., the value function V. However, due to the supervisory data TD target $R + \gamma\hat{Q}(a', s', w)$ already contains the need to optimize parameter w, this is called a semi-gradient, and the gradient is not the fastest direction in decline. 2) For the iterative update of the TD mode since the TD target contains Q that require estimation, which is a biased estimate containing uncertain factors. 3) For the off-policy algorithm control process, two different policies are used during the exploration and optimization, which will make the estimate of the value function inaccurate. The above three factors significantly increase the model instability. Furthermore, for the nonlinear fitting problem (such as NNs), whether MC or SARSA and Q-Learning, it is difficult for these three algorithms to guarantee an optimal solution [64]. Recent algorithms' research has reduced the earlier impact problems, i.e., PPO and SAC algorithms. However, ensuring the model's stability is still one of the future research directions.

3) Predicting Vehicular Mobility: Among the existing issues in the vehicular task offloading environment, mobility of vehicles also causes some challenges, including unreliable wireless link connectivity, short-lived connection, and out-of-range issues. Moreover, in vehicular networks, few RSUs are available to provide services to vehicles, and a situation occurs in no time when a large number of vehicles require URLLC service. As a result, there may be problems with scalability and maintaining the requested users' performance. Similarly, variable vehicle density coupled with fast and frequent topology changes can also cause high packet losses. Specifically, in a multi-hop transmission system, the probability of success is low, considering the connectivity of nodes involved in a highly dynamic environment. One solution to these challenges is to leverage AI tools to predict the vehicle's next position, increasing task offloading efficiency and reducing unnecessary resource consumption. Hence, there is a dire need for a mobility prediction model better to adapt the offloading processes to vehicles' mobility.

4) Task Migration: Computing tasks are migrated between different edge servers. On the one hand, task migration can ensure that tasks are sent to the vehicle from the edge server closest to the vehicle due to the vehicles' mobility. On the other hand, it can be used for load balancing between edge servers, reducing user service delay. Task migration requires virtualization technology, such as VM migration technology or Docker container [138]. Virtualization requires separating the underlying physical transmission of resources, including virtual hardware, operating systems, and networks. The virtualization of the execution environment can improve the system's security [139], and can also realize the requirements of resource sharing, resource aggregation, resource simulation, and resource isolation. Nevertheless, at the same time, virtualization requires more energy, which seems to conflict with reducing resource consumption in task offloading. Today's DNN models can have many parameters; for example, a pre-training YOLO model [140] is approximately 200MB, which has to consider delayed problems. On the other hand, how to realize the virtual migration of tasks on the vehicle is also a big challenge. Moreover, the vehicle's limited computing and storage resources will increase the task delay while realizing the virtual migration, so a critical question of which parts of the model need to be migrated. There are three migration methods [63]: cold migration, warm migration, and live migration. Cold migration is easy to understand and easy to implement, but it is inefficient [141]. Live migration is more efficient [142] since VM does not need to shut down while migration. However, there are more technical challenges with live migration.

5) Security and Privacy Issues: Like any other network, security and privacy are the major concerns for vehicular task offloading. For every system and network, availability, integrity, authenticity, confidentiality, and non-repudiation are the main security properties [143]. Specifically, there are BS, RSUs, and vehicles for vehicle task offloading environments, which exchange some personal information like vehicle's location to provide users with better offloading services. Moreover, untrusted servers/vehicles can get sensitive data during computation offloading. Coupled with the non-existence of a centralized controller, it can further complicate the creation of integrated security and privacy policies, which may pose a potential threat to the security and privacy of users [144]. One of the solutions is the encrypted data or proper authentication and authorization certificates, which can compromise offloading efficiency and QoE because of additional delay or computational power. There is another risk that RSU/BS-based MEC servers can deny service to users due to virus attacks, like Distributed Denial of Service (DDoS), and may lead to offloading failure. Such service interruptions can also affect network resource utilization. Security and privacy strategies are hard to implement for EC and VC systems, lacking centralized and effective management. Therefore, it is necessary to analyze the trade-off between effectiveness and privacy policies due to vehicles' limited computing power. Therefore, privacy and security issues should not be underestimated and require more in-depth research.

VII. Conclusion

This paper presents a comprehensive survey on RL/DRL-based vehicular task offloading in the emerging era of 6G networks. We first lay the fundamental concepts of CC, EC, and VC computing. Then, we define the RL and DRL's basic knowledge. Furthermore, we classify relevant literature according to the specific RL and DRL algorithms and discuss their strengths and shortcomings. Then, we categorize the literature into two main domains based on RL/DRL algorithms, including RL/DRL solutions for edge node and vehicle cloudlet. Edge nodes are further elaborated into three types: MEC server, nearby vehicles, and HMEC. Based on these edge node types under RL/DRL solutions category, we further classify the schemes into value-based and policy-based sub-categories. Finally, we identify several open issues and challenges regarding future performance improvements. We believe that our comprehensive RL/DRL based vehicular task offloading review can provide new insights and promote the development of advance and practical schemes.

References

[1] J. Guo, B. Song, Y. He, F. R. Yu, and M. Sookhak, "A survey on compressed sensing in vehicular infotainment systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2662–2680, May. 2017.

[2] E. Uhlemann, "Connected-vehicles applications are emerging [connected vehicles]," *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 25–96, Mar. 2016.

[3] P. K. Singh, S. K. Nandi, and S. Nandi, "A tutorial survey on vehicular communication state of the art, and future research directions," *Vehicular Communications*, vol. 18, p. 100164, Aug. 2019.

[4] A. Boukerche and V. Soto, "Computation offloading and retrieval for vehicular edge computing: algorithms, models, and classification," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, Aug. 2020.

[5] M. A. Khan, "A survey of computation offloading strategies for performance improvement of applications running on mobile devices," *Journal of Network and Computer Applications*, vol. 56, pp. 28–40, Oct. 2015.

[6] Y. Liu, S. Wang, Q. Zhao, S. Du, A. Zhou, X. Ma, and F. Yang, "Dependency-aware task scheduling in vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4961–4971, Feb. 2020.

[7] F. Arena and G. Pau, "An overview of vehicular communications," *Future Internet*, vol. 11, no. 2, p. 27, Feb. 2019.

[8] J. Wang, C. Jiang, K. Zhang, T. Q. S. Quek, Y. Ren, and L. Hanzo, "Vehicular sensing networks in a smart city: Principles, technologies and applications," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 122–132, 2018.

[9] H. Vahdat-Nejad, A. Ramazani, T. Mohammadi, and W. Mansoor, "A survey on context-aware vehicular network applications," *Vehicular Communications*, vol. 3, pp. 43–57, Jan. 2016.

[10] A. Boukerche and E. Robson, "Vehicular cloud computing: Architectures, applications, and mobility," *Computer networks*, vol. 135, pp. 171–189, Apr. 2018.

[11] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, Apr. 2017.

[12] H. El-Sayed and M. Chaqfeh, "Exploiting mobile edge computing for enhancing vehicular applications in smart cities," *Sensors*, vol. 19, no. 5, p. 1073, Jan. 2019.

[13] X. Hou, Z. Ren, J. Wang, W. Cheng, Y. Ren, K.-C. Chen, and H. Zhang, "Reliable computation offloading for edge-computing-enabled software-defined iov," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7097–7111, 2020.

[14] I. A. Abbasi and A. Shahid Khan, "A review of vehicle to vehicle communication protocols for VANETs in the urban environment," *Future Internet*, vol. 10, no. 2, pp. 14–28, Feb. 2018.

[15] C. Silva, L. Silva, L. Santos, J. Sarubbi, and A. Pitsillides, "Broadening understanding on managing the communication infrastructure in vehicular networks: Customizing the coverage using the delta network," *Future Internet*, vol. 11, no. 1, pp. 1–19, Jan. 2019.

[16] C. R. Storck and F. Duarte-Figueiredo, "A survey of 5g technology evolution, standards, and infrastructure associated with vehicle-to-everything communications by internet of vehicles," *IEEE Access*, vol. 8, pp. 117 593–117 614, June 2020.

[17] W. U. Khan, X. Li, A. Ihsan, M. A. Khan, V. G. Menon, and M. Ahmed, "Noma-enabled optimization framework for next-generation small-cell iov networks under imperfect sic decoding," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2021.

[18] K. Ganesan, J. Lohr, P. B. Mallick, A. Kunz, and R. Kuchibhotla, "NR sidelink design overview for advanced V2X service," *IEEE Internet of Things Magazine*, vol. 3, no. 1, pp. 26–30, Apr. 2020.

[19] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1472–1514, 2020.

[20] Z. Liu, Z. Li, K. Wu, and M. Li, "Urban traffic prediction from mobility data using deep learning," *Ieee network*, vol. 32, no. 4, pp. 40–46, Aug. 2018.

[21] F. Lin, Y. Xu, Y. Yang, and H. Ma, "A spatial-temporal hybrid model for short-term traffic prediction," *Mathematical Problems in Engineering*, vol. 2019, Jan. 2019.

[22] E. Walraven, M. T. Spaan, and B. Bakker, "Traffic flow optimization: A reinforcement learning approach," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 203–212, Jun. 2016.

[23] E. D'Andrea, P. Ducange, B. Lazzerini, and F. Marcelloni, "Real-time detection of traffic from twitter stream analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2269–2283, Mar. 2015.

[24] M. Zhu, Y. Wang, Z. Pu, and et al., "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102662, Aug. 2020.

[25] Y. Xie, F. Li, Y. Wu, S. Yang, and Y. Wang, "D¡sup¿3¡/sup¿-guard: Acoustic-based drowsy driving detection using smartphones," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Jun. 2019, pp. 1225–1233.

[26] J. M. Celaya-Padilla and et al., ""texting & driving" detection using deep convolutional neural networks," *Applied Sciences*, vol. 9, no. 15, p. 2962, Aug. 2019.

[27] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for internet of vehicles," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 12–18, Jul. 2019.

[28] H. Ji, O. Alfarraj, and A. Tolba, "Artificial intelligence-empowered edge of vehicles: Architecture, enabling technologies, and applications," *IEEE Access*, vol. 8, pp. 61 020–61 034, Mar. 2020.

[29] A. Haydari and Y. Yılmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2022.

[30] W. Yu, F. Liang, X. He, Hatcher, and et al., "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, Nov. 2017.

[31] D. Xu, Y. Li, X. Chen, J. Li, P. Hui, S. Chen, and J. Crowcroft, "A survey of opportunistic offloading," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2198–2236, Feb. 2018.

[32] S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, "A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions," *Wireless Communications and Mobile Computing*, vol. 2019, Feb. 2019.

[33] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. lei Zhang, "Vehicular edge computing and networking: A survey," *Mob. Networks Appl.*, vol. 26, no. 3, pp. 1145–1168, Jul. 2021.

[34] R. A. Dziyauddin, D. Niyato, and et al., "Computation offloading and content caching delivery in vehicular edge computing: A survey," *arXiv preprint arXiv:1912.07803*, Dec. 2019.

[35] R. A. Dziyauddin, D. Niyato, N. C. Luong, M. A. M. Izhar, M. Hadhari, and S. M. Daud, "Computation offloading and content caching delivery in vehicular edge computing: A survey," *arXiv*, vol. abs/1912.07803, pp. arXiv–1912, Dec. 2019.

[36] 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. 198 214–198 243, Oct. 2020.

[37] A. Shakarami, M. Ghobaei-Arani, M. Masdari, and M. Hosseinzadeh, "A survey on the computation offloading approaches in mobile edge/cloud computing environment: A stochastic-based perspective," *Journal of Grid Computing*, vol. 18, no. 4, p. 639–671, Dec. 2020.

[38] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 869–904, Jan. 2020.

[39] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Computer Networks*, vol. 182, p. 107496, Dec. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620311634

[40] K. Tan, D. Bremner, J. L. Kernec, and M. Imran, "Federated machine learning in vehicular networks: A summary of recent applications," in *2020 International Conference on UK-China Emerging Technologies (UCET)*, Aug. 2020, pp. 1–4.

[41] R. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, "The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds," *Future Gener. Comput. Syst.*, vol. 28, no. 6, pp. 861–870, Jun. 2012.

[42] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandić, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[43] Z. Ning, F. Xia, N. Ullah, X. Kong, and X. Hu, "Vehicular social networks: Enabling smart mobility," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 16–55, May. 2017.

[44] P. Mell and T. Grance, "The nist definition of cloud computing," Sept. 2011.

[45] Z. Zhou, C. Gao, C. Xu, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Social big-data-based content dissemination in internet of vehicles," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 768–777, Jul. 2018.

[46] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5g: A survey on mec-based approaches to provisioning and flexibility," *IEEE Communications Surveys Tutorials*, vol. 23, no. 1, pp. 596–630, Nov. 2021.

[47] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[48] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza, and W. U. Khan, "Task offloading and resource allocation for iov using 5g nr-v2x communication," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[49] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[50] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. ZHANG, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[51] C. Chen, T. Qiu, J. Hu, Z. Ren, Y. Zhou, and A. K. Sangaiah, "A congestion avoidance game for information exchange on intersections in heterogeneous vehicular networks," *J. Netw. Comput. Appl.*, vol. 85, pp. 116–126, 2017.

[52] F. Lin, L. Yang, K. Xiong, and X. Gong, "Recent advances in cloud-aware mobile fog computing," *Wirel. Commun. Mob. Comput.*, vol. 2019, Jan. 2019.

[53] M. Abuelela and S. Olariu, "Taking vanet to the clouds," in *MoMM*, Nov. 2010.

[54] E. Skondras, A. Michalas, and D. D. Vergados, "Mobility management on 5g vehicular cloud computing systems," *Vehicular Communications*, vol. 16, pp. 15–44, Apr. 2019.

[55] S. S. Manvi and S. Tangade, "A survey on authentication schemes in VANETs for secured communication," *Vehicular Communications*, vol. 9, pp. 19–30, Jul. 2017.

[56] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, in *The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds*, Mar. 2013.

[57] S. Arif, S. Olariu, J. Wang, G. Yan, W. Yang, and I. Khalil, "Datacenter at the airport: Reasoning about time-dependent parking lot occupancy," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2067–2080, Nov. 2012.

[58] N. Vignesh, R. Shankar, S. Sathyamoorthy, and V. Rajam, "Value added services on stationary vehicular cloud," in *ICDCIT*, Feb. 2014.

[59] S. Li and R. Li, "Task allocation based on task deployment in autonomous vehicular cloud," in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, Jul. 2019, pp. 450–454.

[60] X. Huang, R. Yu, J. Liu, and L. Shu, "Parked vehicle edge computing: Exploiting opportunistic resources for distributed mobile applications," *IEEE Access*, vol. 6, pp. 66 649–66 663, Nov 2018.

[61] M. Eltoweissy, S. Olariu, and M. Younis, "Towards autonomous vehicular clouds," in *International Conference on Ad Hoc Networks*. Victoria, BC, Canada: Springer, Aug. 2010, pp. 1–16.

[62] G. Yan, D. Wen, S. Olariu, and M. C. Weigle, "Security challenges in vehicular cloud computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 284–294, Mar. 2013.

[63] S. Olariu, "A survey of vehicular cloud research: Trends, applications and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2648–2663, Jun. 2020.

[64] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[65] A. B. D. Souza, P. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. R. Filho, J. N. D. Souza, V. Chamola, V. H. C. de Albuquerque, and B. Sikdar, "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 198 214–198 243, 2020.

[66] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[67] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *NIPS*, Jun. 2017.

[68] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, May. 1992.

[69] I. Goodfellow, Y. Bengio, and A. C. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, May. 2015.

[70] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84 – 90, Dec. 2012.

[71] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[72] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning." *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/nature/nature518.html#MnihKSRVBGRFOPB15

[73] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, Dec. 2013.

[74] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous internet of things: Model, applications and challenges," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1722–1760, Aug. 2020.

[75] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *ArXiv*, vol. abs/1509.06461, 2016.

[76] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, *An Introduction to Deep Reinforcement Learning*. Computer Science, Nov. 2018.

[77] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, May. 1992.

[78] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. D. Freitas, "Sample efficient actor-critic with experience replay," *ArXiv*, vol. abs/1611.01224, Nov. 2016.

[79] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *ArXiv*, vol. abs/1602.01783, Feb. 2016.

[80] 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," *ArXiv*, vol. abs/1812.05905, Dec. 2018.

[81] T. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, Sept. 2015.

[82] M. Sheraz, M. Ahmed, X. Hou, Y. Li, D. Jin, Z. Han, and T. Jiang, "Artificial intelligence for wireless caching: Schemes, performance, and challenges," *IEEE Communications Surveys Tutorials*, vol. 23, no. 1, pp. 631–661, Feb. 2021.

[83] L. Buşoniu, R. Babuka, and B. D. Schutter, "Multi-agent reinforcement learning: An overview." Springer Berlin Heidelberg, 2010, pp. 183–221.

[84] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings 10th International Conference on Machine Learning Proceedings (ICML-93)*. San Francisco (CA): Morgan Kaufmann, 1993, pp. 330–337.

[85] J. A. Clouse, "Learning from an automated training," in *Working Notes Workshop on Agents that Learn from Other Agents, 12th International Conference on Machine Learning (ICML-95)*, Tahoe City, US, 1996.

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

[87] J. R. Kok, M. T. J. Spaan, and N. A. Vlassis, "Non-communicative multi-robot coordination in dynamic environments," *Robotics Auton. Syst.*, vol. 50, pp. 99–114, Feb. 2005.

[88] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning," *ArXiv*, vol. abs/1706.05296, 2018.

[89] T. Rashid, M. Samvelyan, C. S. D. Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, pp. 178:1–178:51, Mar. 2020.

[90] R. Powers and Y. Shoham, "New criteria and a new algorithm for learning in multi-agent systems," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2005.

[91] A. Aliyu, A. H. Abdullah, O. Kaiwartya, Y. Cao, J. Lloret, N. Aslam, and U. M. Joda, "Towards video streaming in IoT environments: Vehicular communication perspective," *Computer Communications*, vol. 118, pp. 93–119, Mar. 2018.

[92] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[93] T. Refaat, B. Kantarci, and H. Mouftah, "Virtual machine migration and management for vehicular clouds," *Veh. Commun.*, vol. 4, pp. 47–56, Apr. 2016.

[94] L. T. Tan, R. Q. Hu, and L. Hanzo, "Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3086–3099, Apr. 2019.

[95] Z. Ning, P. Dong, and et al., "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 6, pp. 1–24, Dec. 2019.

[96] Z. Ning, P. Dong, X. Wang, and et al., "When deep reinforcement learning meets 5g-enabled vehicular networks: A distributed offloading framework for traffic big data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1352–1361, Feb. 2020.

[97] P. Dai, Z. Hang, K. Liu, X. Wu, and et al., "Multi-armed bandit learning for computation-intensive services in mec-empowered vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7821–7834, Jul. 2020.

[98] M. Ibrar, A. Akbar, S. R. Jan, M. Jan, L. Wang, H. Song, and N. Shah, "Artnet: Ai-based resource allocation and task offloading in a reconfigurable internet of vehicular networks," Dec. 2020, pp. 1–1.

[99] K. Xiong, S. Leng, C. Huang, C. Yuen, and Y. L. Guan, "Intelligent task offloading for heterogeneous v2x communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2226–2238, Apr. 2021.

[100] M. Khayyat, I. A. Elgendy, and et al., "Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks," *IEEE Access*, vol. 8, pp. 137 052–137 062, Jul. 2020.

[101] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang, and X. Shen, "Joint ran slicing and computation offloading for autonomous vehicular networks: A learning-assisted hierarchical approach," *IEEE Open Journal of Vehicular Technology*, vol. 2, pp. 272–288, Jun. 2021.

[102] Y. Ouyang, "Task offloading algorithm of vehicle edge computing environment based on dueling-dqn," in *Journal of Physics: Conference Series*, vol. 1873, no. 1. IOP Publishing, Apr. 2021, p. 012046.

[103] Z. Ning, K. Zhang, and et al., "Joint computing and caching in 5g-envisioned internet of vehicles: A deep reinforcement learning-based traffic control system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 1–12, Aug. 2020.

[104] H. Ke, J. Wang, and et al., "Deep reinforcement learning-based adaptive computation offloading for mec in heterogeneous vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.

[105] W. Zhan, C. Luo, and et al., "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5449–5465, Mar. 2020.

[106] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation offloading in

iot," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9763–9773, Jun. 2021.

[107] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.

[108] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2416–2428, Mar. 2020.

[109] ——, "Multi-agent reinforcement learning based resource management in mec- and uav-assisted vehicular networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 131–141, Jan. 2021.

[110] S. Yu, Q. Liu, and X. Li, "Full velocity difference and acceleration model for a car-following theory," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, pp. 1229–1234, May. 2013.

[111] S. Abdelhamid, H. S. Hassanein, and G. Takahara, "Vehicle as a resource (vaar)," *IEEE Network*, vol. 29, no. 1, pp. 12–17, Jan. 2015.

[112] D. Tang, X. Zhang, M. Li, and X. Tao, "Adaptive inference reinforcement learning for task offloading in vehicular edge computing systems," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[113] J. Shi, J. Du, J. Wang, and J. Yuan, "Distributed v2v computation offloading based on dynamic pricing using deep reinforcement learning," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

[114] C. Chen, Y. Zhang, Z. Wang, S. Wan, and Q. Pei, "Distributed computation offloading method based on deep reinforcement learning in icv," *Appl. Soft Comput.*, vol. 103, p. 107108, May. 2021.

[115] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 190–10 203, Nov. 2018.

[116] K. Zhang, Y. Zhu, and et al., "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7635–7647, Oct. 2019.

[117] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 158–11 168, Nov. 2019.

[118] 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, Jan. 2020.

[119] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9637–9650, Oct. 2020.

[120] J. Zhang, H. Guo, and J. Liu, "Adaptive task offloading in vehicular edge computing networks: a reinforcement learning based scheme," *Mobile Networks and Applications*, vol. 25, no. 5, pp. 1736–1745, Jun. 2020.

[121] K. Wang, X. Wang, X. Liu, and A. Jolfaei, "Task offloading strategy based on reinforcement learning computing in edge computing architecture of internet of vehicles," *IEEE Access*, vol. 8, pp. 173 779–173 789, Sept. 2020.

[122] H. Maleki, M. Başaran, and L. Durak-Ata, "Reinforcement learning-based decision-making for vehicular edge computing," in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, Jun. 2021, pp. 1–4.

[123] Y. Ren, X. Yu, X. Chen, S. Guo, and X.-S. Qiu, "Vehicular network edge intelligent management : A deep deterministic policy gradient approach for service offloading decision," *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 905–910, Jun. 2020.

[124] X. He, H. Lu, M. Du, Y. Mao, and K. Wang, "Qoe-based task offloading with deep reinforcement learning in edge-enabled internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 1–10, Apr. 2020.

[125] Y. Shibata, A. Sakuraba, G. Sato, and N. Uchida, "IoT based wide area road surface state sensing and communication system for future safety driving," in *International Conference on Advanced Information Networking and Applications*. Matsue, Japan: Springer, Mar. 2019, pp. 1123–1132.

[126] D. Jia and D. Ngoduy, "Enhanced cooperative car-following traffic model with the combination of V2V and V2I communication," *Transportation Research Part B: Methodological*, vol. 90, pp. 172–191, Aug. 2016.

[127] G. Naik, B. Choudhury, and J.-M. Park, "IEEE 802.11 bd & 5G NR V2X: Evolution of radio access technologies for V2X communications," *IEEE Access*, vol. 7, pp. 70 169–70 184, May 2019.

[128] Z. Jiang, S. Zhou, X. Guo, and Z. Niu, "Task replication for deadline-constrained vehicular cloud computing: Optimal policy, performance analysis, and implications on road traffic," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 93–107, Feb. 2018.

[129] F. Sun, N. Cheng, S. Zhang, H. Zhou, L. Gui, and X. Shen, "Reinforcement learning based computation migration for vehicular cloud computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018, pp. 1–6.

[130] Z. Wang, Z. Zhong, and M. Ni, "Application-aware offloading policy using smdp in vehicular fog computing systems," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, May. 2018, pp. 1–6.

[131] C. Lin, D. Deng, and C. Yao, "Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3692–3700, Oct. 2018.

[132] 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 Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.

[133] H. Liang, X. Zhang, X. Hong, Z. Zhang, M. Li, G. Hu, and F. Hou, "Reinforcement learning enabled dynamic resource allocation in internet of vehicles," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 1–1, Jul. 2021.

[134] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4192–4203, May. 2019.

[135] S. Lee and S. Lee, "Poster abstract: Deep reinforcement learning-based resource allocation in vehicular fog computing," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May. 2019, pp. 1029–1030.

[136] S. S. Lee and S. Lee, "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 450–10 464, Oct. 2020.

[137] W. Tong, A. Hussain, W. X. Bo, and S. Maharjan, "Artificial intelligence for vehicle-to-everything: A survey," *IEEE Access*, vol. 7, pp. 10 823–10 843, Jun 2019.

[138] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pp. 1–13, Oct. 2017.

[139] D. C. Marinescu, *Cloud computing: theory and practice*. Morgan Kaufmann, May. 2017.

[140] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, Jul. 2017.

[141] D. Marinescu, A. Paya, J. Morrison, and S. Olariu, "An approach for scaling cloud resource management," *Cluster Computing*, vol. 20, pp. 909–924, Nov. 2016.

[142] D. Kapil, E. Pilli, and R. Joshi, "Live virtual machine migration techniques: Survey and research challenges," *2013 3rd IEEE International Advance Computing Conference (IACC)*, pp. 963–969, Feb. 2013.

[143] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, Jun. 2019.

[144] R. Trimananda, A. Younis, B. Wang, B. Xu, B. Demsky, and G. Xu, "Vigilia: Securing smart home edge computing," in *IEEE/ACM Symposium on Edge Computing (SEC)*. Seattle, WA, USA: IEEE, Oct. 2018, pp. 74–89.