A Collaborative Computation and Offloading for Compute-Intensive and Latency-Sensitive Dependency-Aware Tasks in Dew-Enabled Vehicular Fog Computing: A Federated Deep Q-Learning Approach

Kaushik Mishra, *Member, IEEE*, Goluguri N. V. Rajareddy, Umashankar Ghugar, *Member, IEEE*, Gurpreet Singh Chhabra, and Amir H. Gandomi, *Senior Member, IEEE*

***Abstract*—The demand for vehicular networks is prolifically emerging as it supports advancing in capabilities and qualities of vehicle services. However, this vehicular network cannot solely carry out latency-sensitive and compute-intensive tasks, as the slightest delay may cause any catastrophe. Therefore, fog computing can be a viable solution as an integration to address the aforementioned challenges. Moreover, it complements Cloud computing as it reduces the incurred latency and ingress traffic by shifting the computing resources to the edge of the network. This work investigated task offloading methods in Vehicular Fog Computing (VFC) networks and proposes a Federated learning-supported Deep Q-Learning-based (FedDQL) technique for optimal offloading of tasks in a collaborative computing paradigm. The proposed offloading method in the VFC network performs computations, communications, offloading, and resource utilization considering the latency and energy consumption. The trade-offs between latency and computing and communication constraints were considered in this scenario. The FedDQL scheme was validated for dependent task sets to analyze the efficacy of this method. Finally, the results of extensive simulations provide evidence that the proposed method outperforms others with an average improvement of 49%, 34.3%, 29.2%, 16.2% and 8.21%, respectively.**

***Index Terms*—Deep reinforcement learning, federated learning, mobile fog computing, q-learning, vehicular fog computing, task dependency, task offloading.**

I. Introduction

VEHICULAR networks play a consequential part in smart transportation systems due to the rapid evolvement of the Internet of Things (IoT). These vehicular networks facilitate numerous advanced yet complex applications, such as automatic driving, crash detection, AR&VR-enabled intelligent applications, and other interactive modules for passengers. These applications need intensive computations on resources and interactive communications, which are critical challenges for vehicular networks requiring rich complex services. Besides, the ingress traffic on the road makes the computations/communications difficult with the limited capabilities of vehicles. These applications on vehicles are required to handle latency-sensitive data without delay, for which the network connectivity must be stable and accelerated to handle such tasks within a deterministic span of time. However, the slightest delay in communication between vehicles may cause a catastrophe. Therefore, the vehicular network cannot solely be responsible for these latency-sensitive and compute-intensive tasks, thus a high-end computing paradigm-based vehicular network is required.

Cloud computing has been viewed as a viable solution to address these issues for vehicular networks [1], [2]. On this Cloud-based vehicular network, tasks are offloaded to the Cloud for computation and storage due to the high computing capabilities of Cloud VMs. However, the existence of the physical gap between the vehicles and the Cloud server results in a significant latency gap, which reduces the performance and efficiency of task offloading. Thus, it requires a decentralized architecture and full-fledged paradigm that reduces the incurred latency and copes with the compute- and time-intensive and latency-sensitive tasks requiring complex requirements in complementary with Cloud computing.

Fog-based vehicular computing (VFC) is a cutting-edge computing paradigm to address the innate loopholes of the Cloud-based vehicular network [3], [4]. To tackle the requirements of complex tasks, VFC has been envisioned as a potential solution, providing both on-demand computation and communication resources. Integrating Fog computing with Cloud computing considerably reduces the latency and traffic density as well as improves the users' response time. The computation and communication at/to the Fog computing

Manuscript received 27 February 2023; revised 5 May 2023; accepted 25 May 2023. Date of publication 5 June 2023; date of current version 12 December 2023. Open access funding provided by Óbuda University. The associate editor coordinating the review of this article and approving it for publication was N. Kumar. *(Corresponding author: Amir H. Gandomi.)*

Kaushik Mishra, Goluguri N. V. Rajareddy, Umashankar Ghugar, and Gurpreet Singh Chhabra are with the Department of Computer Science and Engineering, GITAM (Deemed to be University), Visakhapatnam 530045, India (e-mail: kmishra@gitam.edu; ngolugur@gitam.edu; ughugar@ieee.org; gurpreet.kit@gmail.com).

Amir H. Gandomi is with the Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia, and also with the University Research and Innovation Center (EKIK), Óbuda University, 1034 Budapest, Hungary (e-mail: gandomi@uts.edu.au).

Digital Object Identifier 10.1109/TNSM.2023.3282795

work in parallel with the services provided at the edge of the network through the Mobile Edge Computing (MEC) or Mobile Fog Computing (MFC) servers equipped in each Road Side Unit (RSU) and Access Points (APs) [5]. However, these RSUs have much less capacity in terms of power, storage and computation. The computation of high-intensive tasks causes high energy consumption at RSUs. In the VFC network, the Fog servers are coupled with RSUs or deployed as a separate entity. In this work, we consider the decoupling mode of Fog servers due to the flexibility in the computations of tasks with disparate requirements. The Fog layer encompasses both homogeneous and heterogeneous Fog nodes to cope with compute-intensive and latency-sensitive tasks. The computational capabilities of Fog nodes change depending upon the vehicular specifications. In architecture, each RSU keeps in touch with all the Fog servers and their corresponding Fog nodes. At a given time, one Fog server may be connected to several RSUs, but each RSU is only ever connected to one Fog server. An RSU either computes locally or offloads a request to the Fog server for processing when it receives one.

A. Problem Definition

The VFC imposes several challenges. The first and most challenging issue is unstable network connectivity. Internet connectivity is an indispensable part of vehicular networks needed for transmitting any request. Unstable network connectivity may lead to a catastrophic situation. Therefore, a Dew-enabled Internet of Things is facilitated in this architecture.

The second most prominent challenge is the high consumption of energy by the RSUs [6]. A considerable amount of energy is consumed by the RSUs during the offloading of the requests and processing of some requests locally. The incurred consumption is dependent on the quality of the medium connected to the Fog servers and the arrival rate of the requests at the RSUs. Although it processes some requests locally, it also consumes energy due to the disparate specifications and requirements of requests. Hence, there is a need for an optimal association among RSUs and Fog servers to reduce energy consumption.

A third challenge is the dispersion of the loads uniformly across Fog nodes to improve the QoS. Khattak et al. [7] and Kai et al. [8] proposed the integration of the principles of Fog computing with the vehicular ad-hoc network. However, the load balancing among Fog nodes in terms of QoS was not considered. Though the loads are of different configurations, these require to be offloaded uniformly across Fog nodes to prevent overloaded or underloaded conditions. The computation of loads depends on the arrival rate and requirements of each request at the corresponding RSU. The overloaded/underloaded condition arises due to the involvement of the different arrival rates at RSUs. In order to increase QoS (resource usage), the collaboration between RSUs and Fog servers must be established in order to significantly reduce the load imbalance.

The system costs also pose another challenge in these vehicular networks. The communication and computation costs together make the system costs. Earlier, RSUs were powered by renewable energy for the offloading and the MFC servers were generating profits from them. However, since the existing resources have become scarcer due to the rapid ingress of traffic with other challenges nowadays, the vehicles can lease the resources to facilitate the task offloading.

B. Motivation

Deep learning (DL), a subset of the Machine Learning (ML) domain, is a promising solution to address complex problems. AI is an enabler for DL to mimic the learning process. Deep Reinforcement Learning (DRL) combines DL and RL, making use of DL's noncognitive behaviour and RL's ability to make decisions [9]. DRL interacts with the vehicles directly and obtains the optimal scheduling/offloading strategy mapping. Based on the existing literature, several works have used DRL to find the optimal offloading decision. For instance, Yao et al. [9] proposed a hybrid resource allocation strategy for VFC using reinforcement learning with heuristic information. Qu et al. [10] proposed a DMRO algorithm integrating DNN with Q-learning and meta-learning approaches to identify the optimal offloading decisions. Ning et al. [11] devised a resource management algorithm using DRL for VEC. To meliorate the effectiveness of vehicular networks (VNs), Maan and Chaba [12] devised a strategy using a Deep Q-network for offloading. To optimize the total utility of VEC, Liu et al. [13] implemented a combined offloading strategy utilizing Q-learning and DRL together. He et al. [14] proposed an offloading method using DRL to improve the Quality of Experience (QoE) for the Internet of Vehicles. However, load balancing was not considered by the aforementioned strategies while allocating the tasks from vehicles, leading to insignificant resource utilization and latency overhead. The traditional ML methods utilize a central server to collect and process the data. However, the central server gets overloaded with computation and communication overheads. Because data privacy is also an indispensable part of data acquisition, the efficiency and accuracy of ML techniques largely depend on the data's size and the central server's capacity, which exhibit challenges in achieving optimal and accurate results with data confidentiality. In order to address the aforementioned issue of traditional ML and DRL techniques, this work proposes a Federated learning-supported Deep Q-learning offloading strategy that facilitates the uploading of data collaboratively into a global model without sharing the raw data [15].

Our work aimed to analyze the task-offloading strategy in two scenarios: (1) considering cooperative and noncooperative MFC servers, and (2) considering task dependency. In the first case, when a vehicle passes through an RSU, the tasks related to the respective vehicle are independently computed by the MFC server associated with that RSU. On the contrary, when many vehicles offload their tasks to the RSU, the RSU chooses whether to compute such tasks locally or transfer them to the next RSU (located in the direction of the moving vehicle) deployed at Fog servers for computation. Hence, the computation is being performed cooperatively. In the second case, we consider dependent tasks for computation. In vehicular networks, vehicles generate a huge amount of

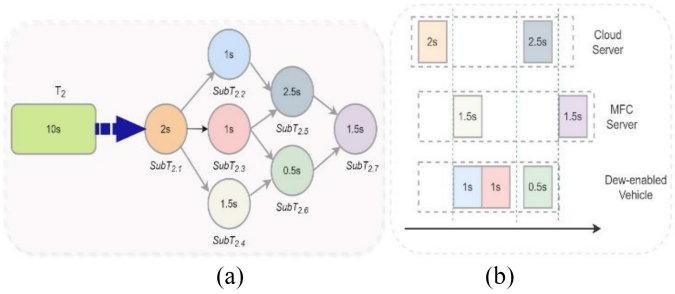

Fig. 1. (a) Representation of a task through a DAG; (b) Fine-grained task offloading.

tasks with heterogeneous requirements that need parallel execution and, hence, are divided into many sub-tasks. Each sub-task is then processed by different computing nodes based on the offloading strategy to reduce both the computation and response time. Figure 1(a) represents the inter-sub-dependency of a task through a Directed Acyclic Graph (DAG), where the task is divided into seven sub-tasks. Figure 1(b) shows the fine-grained task scheduling in the Vehicle-VFC-Cloud collaborative framework.

C. Contribution

As mentioned, this paper proposes a Dew-enabled Federated learning-supported Deep Q-learning-based (FedDQL) offloading method for compute-intensive and latency-sensitive dependent tasks for a collaborative DIoT-Edge-Fog-Cloud framework. This strategy collaboratively optimizes the latency and computations for its intended tasks by reducing the energy consumption of MFC servers. Moreover, the computation loads are uniformly distributed across Fog nodes to prevent load imbalance and, thus, maximize resource utilization. The Fog nodes in the Fog layer are grouped into clusters, each of which contains a homogeneous Fog server, heterogeneous Fog server, Fog nodes, and numerous RSUs. In addition, each cluster has a DQL agent to train the DQL to determine the offloading decisions. A collaborative framework is proposed for tasks offloading in Vehicular Fog Computing. Intermediate nodes, such as routers, gateways, or switches, are deployed on the edge of the network along with a classifier to classify the incoming requests and offload them to the target layers for computation. The Fuzzy classifier is a classifier that determines the target layer for offloading based on the tasks' requirements, including network bandwidth, size in MI, CPU, resource utilization, etc.

The contributions of this paper are delineated as follows:

- Modelling the framework with the principles of Dew computing in order to enable the Internet of Vehicles for uninterrupted task offloading;
- Proposing a collaborative Dew-enabled IoT-Edge-Fog-Cloud computing framework for efficient offloading of tasks, where a DAG depicts the interdependencies among sub-tasks;
- Proposing a Federated learning-supported Deep Q-learning algorithm for dependency-aware task offloading;
- Considering load-balancing, resource utilization, energy consumption, response time (latency), and deadline as QoS metrics to appraise the efficacy of the implemented method; and
- Conducting an empirical assessment of the proposed offloading strategy with other existing methods.

The rest of the paper is organized as follows. Section II reviews the related research. Section III describes the computational models and problem formulation for compute-intensive and latency-sensitive dependency-aware task offloading. The proposed Federated learning-supported Deep Q-learning offloading strategy is elucidated in Section IV. Section V provides an empirical assessment and analysis of the considered QoS parameters for the proposed strategy. Finally, Section VI concludes the paper and highlights future research directions.

II. Related Research

The VANET model has been extended to Vehicular Fog Computing (VFC) by enabling the core principles of Fog computing for the effective offloading of latency-sensitive tasks to compute-intensive vehicles. It facilitates reducing energy consumption, response time, load balancing, and latency while improving resource utilization and, thus, performance. The related research boils down to two primary aspects: (1) Fog computing in vehicular networks, and (2) deep learning for task offloading. The existing works for both of these aspects are briefly summarized as follows.

A. Fog-Assisted Vehicular Networks

A great deal of work has addressed the issues related to vehicular networks using the appealing characteristics of Fog computing in collaboration with the Cloud [16], [17], [18], [19]. For instance, to reduce the total response time of delay-sensitive tasks, the authors of [16] proposed a heuristic-based greedy scheduling algorithm for a three-layered VFC architecture for offloading tasks. Next, a Fog-assisted vehicular computing framework is presented in [17], which is utilized as a testbed for controlling the ingress traffic on the roads. The authors stated that the computations are performed by the RSUs deployed along the road with the assistance of mobile vehicles to meet the demand of the ever-increasing growth of compute-intensive tasks by smart vehicles. To identify congestion among vehicles, [18] proposed a Fog-enabled framework to process the sensed data locally by optimizing the communication. To preserve the privacy of VFC, the authors of [19] introduced a real-time framework called GALAXY supported by Federated learning without the aid of the Cloud.

B. Deep Learning for Task Offloading

In an aim to reduce the transmission power and latency, the authors of [20] devised a collaborative optimization technique to allocate tasks into Fog nodes in an IoT-Fog-Cloud architecture. For efficient resource management, [21] proposed a two-fold technique for allocating tasks and managing vehicular resources optimally in a VFC. In the latter work, contract theory and two-side matching games were formulated for

resource management and task allocation, respectively. Using an actor-critic-based DRL algorithm, the authors of [22] devised an offloading algorithm considering priority in VFC by predicting the dynamic pricing of vehicles. In [4], the authors developed a resource allocation algorithm for parked and slow-moving vehicles in VFC, aiming to reduce parked vehicle service latency. The authors used a combined recurrent neural network (RNN) with DRL and, later, introduced a heuristic strategy to expedite the convergence rate of RL algorithms. To lessen the VM migration and reduce the power consumption during migration, the authors of [23] devised a mechanism called EnTruVe (ENergy and TRUst-aware VM allocation in VEhicular Fog Computing). In addition, an analytical hierarchy process was used to determine an optimal VM based on trust and latency, providing a large pool of vehicular fogs to satisfy the requirements of VMs. In order to minimize the computation-to-communication cost and latency, the authors of [24] proposed a Fuzzy RL-based offloading algorithm that elevates the learning process and maximizes the long-term reward over the Q-learning algorithm. The authors of [25] introduced a dynamic clustering-enabled clustering-based load balancing approach. Moreover, the technique considers the speed, direction, and position of moving vehicles to form groups for making a pool of computing resources. In addition, it also employs a capacity-based load-balancing approach for performing the distribution of loads between vehicles and among the cluster of vehicles. To decide the target layers for offloading, the authors of [26] proposed a Federated learning-supported DRL mechanism for VFC, which learns through the learning model among Fog nodes and vehicles, resulting in fast convergence. Network overhead is also significantly reduced due to the implementation of Federated learning, and thus, the privacy of the users' data is also preserved.

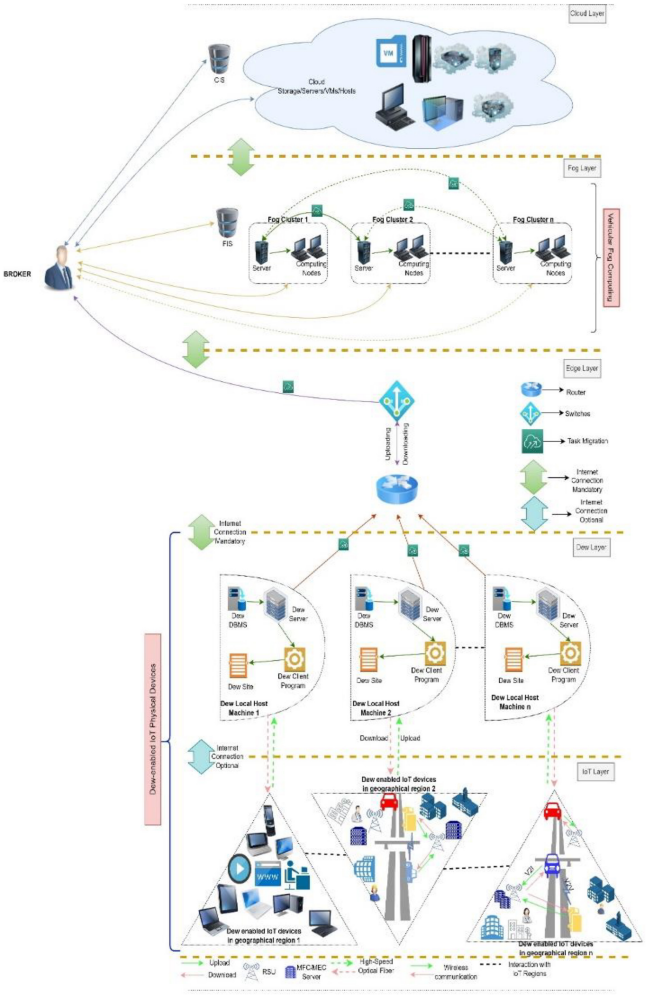

Fig. 2. Dew-enabled task offloading model.

III. System Models and Problem Formulation

Initially, this section introduces a dew-enabled VFC network model for task offloading. Next, different system models, including the vehicular task, communication, and computation models, are elaborated. Furthermore, the problem formulation with key objectives is discussed.

A. Dew-Enabled VFC Network Model

The proposed dew-enabled Vehicular Fog Computing model, a collaborative task offloading network/communication framework, is illustrated in Fig. 2. The model consists of five layers: the bottom IoT layer, Dew layer, Edge layer, Fog layer, followed by the Cloud layer. The roles and significance of each layer in task offloading in a vehicular network are described as follows.

At the ground layer, the IoT layer consists of numerous Internet-enabled physical devices (vehicles), which generate requests of disparate specifications. Vehicles also contain local processing units. All communications take place through a stable Internet connection. However, a stable connection around the clock seems impractical, and any catastrophe may arise due to unstable connectivity. Therefore, the core principles of Dew computing are embedded for uninterrupted communication during unstable connectivity. Dew servers consist of local host machines that contain Dew DBMS, Dew Client program, and Dew interactive sites. Dew servers keep backups in their storage and provide services when unstable connectivity occurs. Meanwhile, if the user updates any data then the Dew server keeps it in the Dew DBMS and makes it reflected on the user program when the connectivity becomes stable. Therefore, these Dew-enabled IoT devices are deployed in the geographical region for uninterrupted services for vehicular networks.

Vehicles with Internet connectivity transmit requests to the Cloud or Fog layer to be processed via the Edge layer. The Edge layer is made up of plenty of intermediary nodes, including routers, switches, or gateways, that are used to direct the requested packets to the following layer immediately. Additionally, it uses a Fuzzy logic classifier to categorize the tasks in order to identify the target offloading layers based on the requirements of each task. However, this layer exhibits some challenges in terms of security, cost, connectivity, scalability and complexity [8], [9], [11]. This concept is theoretical and close to impossible in implementation. This layer is a replica of how the computations can be done on the edge of the network via intermediate nodes. The requests are then

sent to the Cloud or Fog layer, depending on which one will handle them. The Fog layer is an amalgamation of different Fog nodes with homogeneous and heterogeneous resources. These Fog nodes are computing nodes in terms of virtual machines (VMs) to process requests either collaboratively or locally in this layer. Furthermore, these computing nodes are clustered together based on the computing capacity into two groups: Collaborative Fog Nodes (CFNs), which is a collection of both homogenous and heterogeneous resources that are able to handle tasks (or requests) with disparate specifications; and Local Fog Nodes (LFNs), which is a combination of only homogeneous resources that is capable of handling tasks with minimum completion time or requirements. Moreover, this layer also deploys RSUs and MFC servers to communicate among vehicles. Each RSU is embedded with an MFC server and gathers service requests from vehicles within its range while each MFC server processes tasks and subtasks through offloading. Here, the nodes are geographically distributed facilitating the execution of the tasks in a distributed way to leverage the centralized architecture of the Cloud datacentre. The tasks with high requirements or tasks taking longer to execute in the Fog layer are offloaded to the Cloud for processing and storage. The Cloud datacentre encompasses the physical machines (or hosts) and VMs out of hosts through virtualization technology. The VMs are the computing entities for enabling the scheduling and execution of high-end tasks. The centralized architecture of the Cloud datacentre results in high response time and latency overhead. The Fog layer is introduced in the proposed framework to minimize these issues.

We assume that the tasks are dependent in nature and, hence, partitioned into subtasks. Furthermore, we assume that *N* compute-intensive and latency-sensitive tasks are connected with each vehicle and will be scheduled and computed within a deterministic completion time. The number of RSUs is represented as $R = \{1,\ 2,\ 3, \ldots,\ r, \ldots, R\}$, and the group of vehicles is referred to as $V = \{1,\ 2,\ 3, \ldots,\ v, \ldots, V\}$. The variable T_n^v refers to the n^{th} task of the vehicle *v*.

B. Vehicular Task Model

This section introduces dependency-aware task offloading for the vehicular task model. Interdependency among tasks is considered as the tasks are not atomic. In other words, when a task is broken down into subtasks, the processing of each subtask is dependent upon its previous subtask's execution. The processing of each subtask is based on the requirements of each subtask and is offloaded to the respective target layer for processing (locally (vehicle) or via Fog or Cloud). The subtasks of a task are represented through a Directed Acyclic Graph (DAG) and modelled as $G = (\tau,\ \varepsilon)$, where τ is the array of subtasks and ε is the pair of directed edges depicting interdependencies between two subtasks. Let $I = |\tau|$ represent the total set of subtasks of a task T_n^v. In DAG, a vertex $T_{n,i}^v$ denotes the i^{th} subtask of the n^{th} task of the vehicle *v*, and a directed edge $(T_{n,i}^v,\ T_{n,j}^v)$ indicates the interdependency between the i^{th} subtask and j^{th} subtask of T_n^v. The j^{th} subtask can be executed only if the i^{th} subtask has been completed, where $i,\ j \in I$.

As shown in Fig. 1, the task T_2^3 $(n = 1, 2, \ldots n;\ v = 1, 2, \ldots v)$ is broken down into seven subtasks, such as $T_{2,1}^3$, $T_{2,2}^3$, $T_{2,3}^3$, $T_{2,4}^3$, $T_{2,5}^3$, $T_{2,6}^3$, and $T_{2,7}^3$ of the form $T_{n,\ i;i=1,2,\ldots,7}^v$. The subtask $T_{2,\ 1}^3$ is the entry subtask and executes first, and $T_{2,7}^3$ is the exit subtask and executes at the end until all its predecessor subtasks have been executed. Each subtask $T_{n,i}^v$ is defined with four attributes, i.e., $T_{n,i}^v = \langle L_{n,i}^v,\ D_{n,i}^v, d_{n,i}^v, C_{n,i}^v \rangle$, where $\langle L_{n,i}^v \rangle$ denotes the length of the subtask in MI, $\langle D_{n,i}^v \rangle$ is the latency rate of the subtask, $\langle d_{n,i}^v \rangle$ is the deadline (hard or soft or firm) of the subtask, and $\langle C_{n,i}^v \rangle$ is the number of resources required for computation to execute $T_{n,i}^v$.

In our approach, a task can be categorized into three classes, namely crucial, high-end, and low-end tasks. We assume that the crucial tasks are those which contain some crucial information about the vehicle and, hence, can be computed in the vehicle itself. High-end tasks are associated with some priority and deadline. If the high-end tasks do not meet the deadline, the outcome must not be considered or the task is considered as failed. Low-priority tasks also have some priority and deadlines. Unlike high-priority tasks, if the low-priority tasks do not meet the deadline, the outcome is still in use, but the outcome will not be considered if the execution time increases. If the execution of a task is longer than usual, then it is forwarded to the upper layer for processing. However, the vehicles must ensure the execution of high-priority tasks over low-priority tasks. A utility in terms of reward or penalty is associated with both high-priority and low-priority tasks upon meeting or missing deadlines. For the high-priority task, a log function is used to define the utility of a subtask given as:

$$\mu_n^H = \begin{cases} \log(1 + \tau_n - t_{C_n}),\ t_{C_n} \leq \tau_n \\ -\Gamma^H, \qquad\qquad\ t_{C_n} > \tau_n \end{cases} \tag{1}$$

where t_{C_n} denotes the completion time of a subtask $T_{n,i}^v$; and $-\Gamma^H$ implies a penalty for not being able to meet the deadline.

For the low-priority task, if the completion time of the executing subtask is less than the defined deadline, then the utility is a reward. Otherwise, the utility is a penalty and expressed as:

$$\mu_n^L = \begin{cases} \Gamma^L, \qquad\qquad\ t_{C_n} \leq \tau_n \\ \Gamma^l\ e^{-c}(t_{C_n} - \tau_n),\ t_{C_n} > \tau_n \end{cases} \tag{2}$$

where Γ^L defines a positive reward (constant); and c is a constant and greater than zero.

In order to offload a subtask to any other vehicle (call it a service vehicle), the task vehicle has to pay a unit price denoted as ρ_n. Hence, the computation size of the subtask (C_n) can be estimated as $C_n = f_n t_n'$, where f_n is the frequency assigned to the service vehicle, and t_n' denotes the computing time of the subtask. Finally, the utility of the task vehicle can be expressed as:

$$Util_n = \mathbb{1}(T_n = T_H)\mu_n^H + \mathbb{1}(T_n = T_L)\mu_n^L - \rho_n C_n \tag{3}$$

where $\mathbb{1}(.)$ is an indicator function [28]. and T_H and T_L denote the high-priority and low-priority subtasks, respectively.

TABLE I
TARGET OFFLOADING LAYERS AS INFERENCES USING FUZZY LOGIC

Fuzzy Inputs				Output
Task Size (MI)	**Network Bandwidth (Mbps)**	**Latency-Sensitivity**	**Deadline-constrained**	**Target Layer**
h	*h*	*h*	*h*	$\mathcal{Q}_1$
h	*h*	*h*	*m*	$\mathcal{Q}_1$
h	*h*	*m*	*l*	$\mathcal{Q}_1$
h	*h*	*m*	*h*	$\mathcal{Q}_1$
h	*m*	*l*	*m*	$\mathcal{Q}_2$
h	*m*	*l*	*l*	$\mathcal{Q}_2$
h	*m*	*h*	*h*	$\mathcal{Q}_1$
h	*m*	*h*	*m*	$\mathcal{Q}_1$
m	*l*	*m*	*l*	$\mathcal{Q}_2$
m	*l*	*m*	*h*	$\mathcal{Q}_2$
m	*l*	*l*	*m*	$\mathcal{Q}_2$
m	*l*	*l*	*l*	$\mathcal{Q}_2$
m	*h*	*h*	*h*	$\mathcal{Q}_2$
m	*h*	*h*	*m*	$\mathcal{Q}_2$
m	*h*	*m*	*l*	$\mathcal{Q}_2$
m	*h*	*m*	*h*	$\mathcal{Q}_2$
l	*m*	*l*	*m*	$\mathcal{Q}_3$
l	*m*	*l*	*l*	$\mathcal{Q}_3$
l	*m*	*h*	*h*	$\mathcal{Q}_3$
l	*m*	*h*	*m*	$\mathcal{Q}_3$
l	*l*	*m*	*l*	$\mathcal{Q}_3$
l	*l*	*m*	*h*	$\mathcal{Q}_3$
l	*l*	*l*	*m*	$\mathcal{Q}_3$
l	*l*	*l*	*l*	$\mathcal{Q}_3$

h: high, m: medium, l: low;
$\mathcal{Q}_1$: Cloud Server, $\mathcal{Q}_2$: MFC Server, $\mathcal{Q}_3$: Local Server

C. Classification of Offloading Layers Using Fuzzy Logic

Fuzzy logic is used to determine the target offloading layers for the tasks generated by vehicles. It is implemented to minimize the latency incurred while executing high-end and low-end tasks at one layer. Specifically, Fuzzy logic offloads different groups of tasks requiring disparate computational capabilities of resources to different target layers. It prevents starvation and aging. By this method, if the execution of a task is taking longer than usual, the respective task is forwarded to the next adjacent layer for execution. It is manifested through three modules: Fuzzy inputs, Fuzzification, and Defuzzification. *Fuzzy inputs* are the essential parameters of the Fuzzy logic model that draw the inferences, which include Tasks size (MI), Network Bandwidth (Mbps), Latency-sensitivity, and Deadline-constrained. These inputs are characterized by High (*h*), Medium (*m*), and Low (*l*) lexical parameters representing the heterogeneity and dynamicity of tasks. *Fuzzification* is the process of drawing inferences based on Fuzzy inputs with the aid of an inference engine. Fuzzy Knowledge Base is a repository consisting of inferences in terms of Fuzzy rules. Fuzzy inputs are processed based on the defined membership functions. *Defuzzification* is a process of transforming the Fuzzy inferences to some consolidated values based on membership functions. Table I delineates the required output (target offloading layers) based on Fuzzy inputs.

D. Vehicular Communication Model

This section describes the communications and interactions among different entities for smooth offloading. The interactions refer to the communication model of vehicles to RSUs, from RSUs to the Cloud server, and from vehicles to the Cloud. The communications among these entities are illustrated as follows.

Vehicles to RSUs: The Internet-enabled vehicles (IoV) communicate with RSUs through wireless communication based on Frequency-Division Multiple Access (FDMA). To transmit a byte of request, a vehicle v requires Tr_v^r amount of data transmission rate to an RSU r, expressed in Eq. (4) [27]:

$$Tr_v^r = h_v^r \times bw \times \left(1 + Tp_v \times \beta_v \times \rho^{-2}\right) \tag{4}$$

where h_v^r is the allocated sub-channels to the vehicle v; bw is the allocated network bandwidth for each sub-channel; Tp_v is the power to transmit each byte from the v^{th} vehicle; β_v is the gain of the channel; and ρ^{-2} is the noise in the surrounding environment.

RSUs to Cloud Server: An optical fibre-based wired connection is used to transmit tasks from RSUs to the Cloud server. If the MFC server fails, tasks or subtasks are migrated to the Cloud server for computation. Some transmission latency occurs while offloading to the Cloud. The transmission latency and acknowledgement latency are equivalent between the MFC server and Cloud. While the transmission latency is independent of the lengths of the tasks, it results due to the presence of the physical gap between the MFC and Cloud. Nonetheless, the incurred latency, in this case, is lesser than the direct offloading of tasks from vehicles to the Cloud server. Therefore, we denote the round-trip time (τ_r^{Cloud}) for transmitting data from the RSU to the Cloud and get the ack in Eq. (5):

$$\tau_r^{Cloud} = 2 \times D_{off(r)}^{Cloud} \left[\because\ D_{off(r)}^{Cloud} + D_{ack(Cloud)}^{r}\right] \tag{5}$$

where $D_{off(r)}^{Cloud}$ denotes the transmission latency incurred while offloading a request from RSU r to the Cloud, and $D_{ack(Cloud)}^{r}$ is the latency incurred for sending an *ack* from the Cloud to the RSU r.

Vehicles to Cloud server: The high-end tasks are directly offloaded to the Cloud servers for computation, thus requiring high-end specifications. Therefore, the communication involves the transmission latency of sending a request from a vehicle v to the Cloud server and is expressed in Eq. (6):

$$Tr_v^{Cloud} = h_v^{Cloud} \times bw \times log_2 \times \left(1 + Tp_v \times \beta_v \times \rho^{-2}\right) \times L_i + ack \tag{6}$$

where L_i denotes the length of the i^{th} request; and h_v^{Cloud} is the allocated sub-channels to the vehicle v.

E. Computation Model

This section explains the different computation models in this vehicular Fog computing architecture, which includes the

energy consumption model, latency-aware computation model, and pricing model. Each model is elucidated below.

1) Energy Consumption Model: The total energy consumed at time t when a vehicle transmits a request either to the Fog layer or the Cloud layer via RSU is expressed in Eq. (7):

$$Ec_i(t) = Ec_i^r(t) + Ec_i^{Fog/Cloud}(t) \tag{7}$$

The energy consumed at time t at the RSU r ($Ec_i^r(t)$) while propagating the request from the vehicles is estimated in Eq. (8):

$$Ec_i^r(t) = L_i \times Ec_{nw}^r(i,t) + L_i \times h_i \times \left(\frac{P_{idle}^r}{Cap_{idle}^r} + \frac{P_{\max}^r}{Cap_{\max}^r Util^r} \right) \tag{8}$$

where L_i is the length of the i^{th} request; $Ec_{nw}^r(i,t)$ denotes the energy consumed to transmit each byte of request from a vehicle v to the RSU r via the network; h_i is the transmission gain of the channel; P_{idle}^r and $P_{\max}^r$ are the energy consumed during the idle and active states of the RSU r, respectively; Cap_{idle}^r and $Cap_{\max}^r$ are the capacity of the RSU r during the idle and active states, respectively; and $Util^r$ is the utilization of the RSU r.

Next, the task is either offloaded to the Fog or Cloud for computation. Here, the energy is consumed while processing and storing the requests as well as sending the respective acknowledgement. Hence, the energy is mathematically expressed as follows:

$$Ec_i^{Fog/Cloud}(t) = Ec_i^{P\&S} + \beta \times Ec_i^{Cloud} + Ec_{ack} \tag{9}$$

The energy consumed while processing and storing the request either at Fog node or Cloud VM, $Ec_i^{P\&S}$, is expressed in Eq. (10):

$$Ec_i^{P\&S} = L_i \times Ec_i^P + L_i \times Ec_i^S \tag{10}$$

$$Ec_i^P = \left[\delta \times P_j^{active} + (1-\delta) \times P_j^{idle} \right] \times \frac{St(j,t)}{nT(j,t)} \tag{11}$$

where δ is the ratio ($\frac{T_{active}}{T_{total}}$) of the active time to total time of the j^{th} Fog node or Cloud VM; $(1-\delta)$ denotes the elapsed time for the j^{th} Fog node or Cloud VM; P_j^{active} and P_j^{idle} denote the power consumption of the j^{th} Fog node or Cloud VM during the idle and active states, respectively; $St(j, t)$ is the service time of the j^{th} Fog node or Cloud VM; $nT(j, t)$ is the number of tasks associated with the j^{th} Fog node or Cloud VM; and $\beta(= \frac{T_{req}^{sent}}{T_{req}})$ is the ratio of the total request sent to the total request offloaded to the Cloud from Fog nodes upon failure.

Ec_i^{Cloud} is the energy consumed by the j^{th} Cloud VM when any Fog node fails to execute, as computed in Eq. (12):

$$Ec_i^{Cloud} = Ec_i^{P\&S} + Ec_{ack} \tag{12}$$

Ec_{ack} is the energy consumed while sending the acknowledgement back to RSU or the corresponding Fog node and is expressed in Eq. (13):

$$Ec_{ack} = L_i \times Ec_{nw}^{r/Fog}(i,t) \tag{13}$$

2) Latency-aware Model: A task's subtasks can be carried out locally (on the vehicle itself) or offloaded to a Cloud server or an MFC server at the Fog layer for processing. At various phases of processing and offloading, variable rates of latency might occur. In this instance, the processing times of the subtasks at the vehicles, MFC server, and Cloud server can be used to determine the latency.

Computing on vehicles: When the tasks are classified and offloaded to the target layers for computation, tasks with their subtasks arrive at the queue of the vehicle for computation. We assume that all the tasks in the queue are in FIFO order and the vehicle processes one subtask at a time. That means, while a subtask is in execution, other subtasks must wait in the queue. This processing involves both processing time and waiting time in the queue. Hence, the total latency incurred while computing on vehicles for a subtask $T_{n,i}^v$ is given by Eq. (14):

$$D_{local(n,i)}^v = \frac{C_{n,i}^v}{c^v} + D_{wait(n,i)}^v = \frac{C_{n,i}^v}{c^v} + T_{start(n,i)}^{local(v)} - T_{request(n,i)}^{local(v)} \tag{14}$$

where $\frac{C_{n,i}^v}{c^v}$ denotes the local processing latency; $D_{wait(n,i)}^v$ is the waiting time of a subtask $T_{n,i}^v$; $C_{n,i}^v$ is the number of computing resources needed to execute the subtask $T_{n,i}^v$; c^v is the computation capability for the vehicle v; and $T_{start(n,i)}^{local(v)}$ and $T_{request(n,i)}^{local(v)}$ are the starting and requested times of a subtask $T_{n,i}^v$, respectively.

Computing on MFC server: The latency rate at the MFC server is the sum of the transmission delay from the vehicle to the MFC server via RSU, processing delay, waiting delay of the subtasks on the MFC server to be scheduled, acknowledgement delay back to the vehicle, and the offloading latency to the Cloud upon failure. These factors are formulated as follows:

First, the transmission delay in propagating a request i to the respective Fog node is estimated as:

$$D_{(n,i)}^{fog} = L_i \times T_L^{nw} \tag{15}$$

where T_L^{nw} denotes the transmission delay caused by propagating each byte of request i from the vehicle v to an MFC server $fn^j; j = 1, 2, \ldots, m$.

Second, the latency caused by processing the request i on an MFC server is denoted as:

$$D_P^{fog} = L_i \times \frac{St(j,t)}{nT(j,t)} \tag{16}$$

Third, transmitting an acknowledgement to the respective vehicle v causes some latency and is denoted as:

$$D_{ack(n,i)}^v = L_i \times T_L^{nw} \tag{17}$$

Next, the waiting latency at the queue of the MFC server is similar to the local waiting delay and is expressed as follows:

$$D_{wait(n,i)}^{fog} = T_{start(n,i)}^{fog} - T_{request(n,i)}^{fog} \tag{18}$$

Finally, the transmission delay caused to offload of a request i to one of the Cloud servers upon failure is expressed as

follows:

$$D_{(n,i)}^{Cloud} = 2(L_i \times T_L^{nw}) + L_i \times \frac{St(j,t)}{nT(j,t)} \tag{19}$$

Here, the transmission delay states that when a request is sent from an IoT device to a Cloud server, the time it takes for the request to reach the server (transmission delay) is equivalent to the time it takes for the acknowledgement (response) to travel back to the IoT device. Therefore, the transmission delay is doubled because it includes both the time for the request to reach the server and the time for the response to reach the IoT device. So, the total transmission delay is the original delay multiplied by two.

By combining Eqs. (15)-(19), we get the total latency caused at the MFC server in Eq. (20):

$$D^{fog} = D_{(n,i)}^{fog} + D_P^{fog} + D_{ack(n,i)}^{v} + D_{wait(n,i)}^{fog} + \beta D_{(n,i)}^{Cloud} \tag{20}$$

Computing on Cloud Server: The subtask $T_{n,i}^{v}$ is migrated to the Cloud server after being offloaded from the vehicle v to the RSU r for processing. The transmission latency includes transmission delay, processing delay at the Cloud server, and acknowledgement delay, which are analogous to the MFC server. Hence, the total latency caused by the Cloud server is defined as follows:

$$D^{Cloud} = D_{(n,i)}^{Cloud} + D_P^{Cloud} + D_{ack(n,i)}^{v} \tag{21}$$

3) Pricing Model: In VFC, the low-end tasks are computed on the vehicle itself, and the high-end and medium-end tasks get scheduled on the Cloud server and Fog server due to the lack of computational capabilities of the vehicles. Hence, a fair allocation of resources should adhere to the computation of high-end tasks to meet their deadline on a priority basis. Let's consider T_n number of high-end tasks that are generated from the vehicles V_v. The deadline and computation size of a task/subtask are referred to as d_i and L_i, respectively. The minimum frequency required for the local subtasks is denoted as $f_v^{\min} = \sum_{i=1}^{T} \frac{L_i}{d_i}$. Upon the availability of computing resources, a vehicle might deny offloading the requests. Then, the frequency is computed as $f_v = \sum_{i=1}^{T} \frac{L_i}{\vartheta_v d_i}$, where $\vartheta_v = f_v^{\min}/f_v$ and the reserved frequency is $[f_v^{\min},\ f_v]$. Here, ϑ is used to denote the ratio of minimum and maximum frequencies to compute the subtasks locally, and the range of ϑ is $[\vartheta_v,\ 1]$. Therefore, we denote the total utilization of local subtasks as [22]:

$$Util_{local}^{v}(\vartheta_v) = \sum_{i=1}^{T_n} \log(1 + d_i - \vartheta d_i),\ \vartheta \in [\vartheta_v,\ 1] \tag{22}$$

In one scenario, a vehicle migrates its subtasks to a passing vehicle for computation due to the lack of computational capabilities and allocates frequency f_n for the migrated subtask $T_{n,\ i}^{v}$. Therefore, the utilization of the migrated local subtasks on a new vehicle is denoted as $Util_{local}^{v}(\vartheta'_v)$, and the cost (ρ_n) paid for the subtask $T_{n,\ i}^{v}$ should meet the requirement as $\rho_n = Util_{local}^{v}(\vartheta_v) - Util_{local}^{v}(\vartheta'_v)$. This is compensation for the loss of the utilization of the local subtask. The frequency devoted to the offloaded subtask is estimated as $f_n = f_v - f_v^{\min}/\vartheta'_v$, where ϑ'_v can be estimated from the above requirement condition [22]. Consequently, if there is a need for computing resources in order to compute the offloaded subtasks, then the cost will be higher for the respective vehicle v.

F. Problem Formulation

The objective functions can be formulated as:

$$\min_{T_{n,i}^{v} r_{n,i}^{v}} \sum\sum \beta \times T_{L_{n,\ i}^{v}}(t) + \alpha \times E_c(t),\ r \in R,\ v \in V,\ n \in N; \tag{23}$$

$$\max \frac{1}{N} \sum_{n=1}^{N} \sum_{v=1}^{V} C_n^v \Big(\mathbb{1}(T_n = T_H) U_n^H + \mathbb{1}(T_n = T_L) U_n^L - \rho_n C_n \Big) \tag{24}$$

subject to the following constraints:

$$\text{C1}: \beta + \alpha = 1; \tag{25a}$$

$$\text{C2}: T_{L_{n,\ i}^{v}} = D_{local(n,\ i)}^{v} + D^{fog} + D^{Cloud}; \tag{25b}$$

$$\text{C3}: X_{i,j}(t) \in \{0,\ 1\},\ \forall i \in R,\ \forall j \in V; \tag{25c}$$

$$\text{C4}: \sum_{i \in R} \sum_{j \in V} 0 \le X_{i,j}(t) \le 1; \tag{25d}$$

$$\text{C5}: \sum_{v \in V} Comp_{v,\ m} \le \max_{cap}^{comp},\ Comp_{v,m} \in r_{n,i}^{v}; \tag{25e}$$

$$\text{C6}: \sum_{v \in V} Comm_{v,\ m} \le \max_{cap}^{comm},\ Comm_{v,m} \in r_{n,i}^{v}; \tag{25f}$$

$$\text{C7}: 0 \le \rho_n^v \le \frac{Util_{local}^{v}\left(\widetilde{\vartheta_v}\right)}{C_n},\ \forall n \in N,\ \forall v \in V; \tag{26a}$$

$$\text{C8}: 1(C_n^v = 1) > 0,\ \forall n \in N,\ \forall v \in V; \tag{26b}$$

Eq. (23) indicates two key objectives: (1) reducing the overall computational latency ratio across the computing layers, and (2) minimizing the average energy consumption across RSUs. Eq. (24) implies that the average utilization of offloading subtasks should be maximized.

The constraint $C1$ in Eq. (25a) is the addition of weights $(\beta$ and $\alpha)$ of two sub-objectives $(T_{L_{n,\ i}^{v}}(t)$ and $E_c(t))$ stated in Eq. (23). Eq. (25b) represents the total computational latency incurred while transmitting requests from the RSU to Fog, RSU to Cloud, and then Fog to Cloud. In Eq. (25c), $X_{i,\ j}(t)$ is a binary variable that states the assignment of the subtasks to the desired MFC servers for computation. Eq. (25d) states that the RSU j can be linked with at least one Fog server m, but a Fog server m can be linked with more than zero RSUs at a given instant. In Eq. (25e), $Comp_{v,\ m}$ and $Comm_{v,\ m}$ are the allocated computation and communication resources, respectively, and should not exceed the maximum computation and communication capabilities of MFC servers. The constraint $C7$ guarantees a positive cost value within the range of the maximum value, and $\widetilde{\vartheta_v}$ denotes the current value of ϑ in V_v while offloading a task. The constraint $C8$ denotes the availability of the selected vehicle for the subtask offloading.

This section formalized the task offloading problem in VFC. This optimization problem aims to reduce the total

computational latency and mean energy consumption while maximizing the total utilization of offloading subtasks in terms of cost subject to the constraints of communication and computing resources, the nature of the association between the RSUs and Fog servers, the binary mapping of subtasks to the compatible Fog servers or Cloud servers, and the availability of the selected vehicle to offload the subtask when a vehicle fails to compute. In a dynamic context (unpredictable requests with disparate requirements) where the level of collaboration between RSUs and Fog servers/Cloud servers fluctuates, it is challenging to optimize the objective functions in Equations 23 and 24 using conventional optimization approaches like heuristics or metaheuristics. In other words, it is not easy to find the best solutions for the objectives given the constantly changing level of collaboration between the servers and RSUs due to the unpredictable demands. Hence, the next section proposes a Federated learning-supported Deep Q-learning offloading (FedDQL) strategy to optimize the aforementioned objective functions.

IV. Task Offloading Algorithm: A Federated Learning (FL)-Supported Deep Q-Learning (FedDQL) Approach

This section first discusses the conventional Deep Q-learning neural networks for identifying a dynamic vehicular offloading in the Fog paradigm. Afterwards, a Federated Learning (FL)-supported Deep Q-learning strategy for global offloading is presented. In the proposed offloading strategy, the conventional Q-learning is amalgamated with deep neural networks as the proposed DQN strategy. In this network, an agent acts by learning from the environment and optimizes the sum of rewards. This environment, in terms of the VFC environment, can be formulated as a Markov Decision Process expressed as *env* : $\{S(t), Ac(t), Re(t), P(t)\}$, where $S(t)$ denotes the state space of the environment at time t, $Ac(t)$ implies the action at time t performed by the DQN agent, $Re(t)$ expresses the reward space an agent gets, and $P(t)$ is the probability of transition. Here, an agent indicates the vehicle associated with the number of RSUs and Fog/Cloud servers in the Vehicle-Fog-Cloud environment. Each terminology used in the environment is illustrated as follows.

State: A state space of an environment is comprised of computational models in terms of mean energy consumption of RSUs and the computational latency incurred while transmitting request i from a vehicle v and is defined as [11]:

$$S(t) = \{Ec(1,\ t),\ Ec(2,\ t), \ldots,\ Ec(|Z|,\ t), \\ D(1,t),\ D(2,t), \ldots, D(|H|,t)\},\ i \in Z,\ j \in H \quad (27)$$

where $Ec(i, t)$ is the energy consumption by the i^{th} RSU at time t; and $D(j, t)$ is the computational latency incurred by the j^{th} Fog server at time t.

Action: An action space comprises an action in terms of offloading the requests by an agent. Hence, the action $Ac(t)$ of a subtask i of the task $T^v_{n,\ i}$ at time t is given by [11]:

$$Ac(t) = \{X_{i,\ j}(t),\ i \in Z,\ j \in H; \\ Alloc_{comp}(t),\ Alloc_{comm}(t)\} \quad (28)$$

where $X_{i,\ j}(t)$ is the mapping of subtask i to a compatible the j^{th} Fog/Cloud server; and $Alloc_{comp}(t)$ and $Alloc_{comm}(t)$ are the allocated computation and communication resources for the subtask i of the task $T^v_{n,\ i}$, respectively.

Reward: Based on the execution of the action on the state defined in the formulated environment, a reward is facilitated. The reward $Re(t)$ is the negation of the weighted total of both average energy consumption and computational latency at time t given as [11]:

$$Re(t) = \left\{-\left(\beta \times T^v_{Ln,\ i}(t) + \alpha \times E_C(t)\right)\right\} \quad (29)$$

An agent must select the action $Ac(t)$ as an offloading of the subtask i at state $S(t)$ to get the optimal reward while minimizing the total energy consumption across RSUs and the computational latency. In the next sections, the conventional Q-learning approach is demonstrated, followed by the proposed FedDQL offloading strategy.

A. Standard Q-Learning Approach

The Q-learning approach is used to identify the optimal decision-making policy through a function $\mathcal{Q}\{S(t) \rightarrow Ac(t)\}$. It is a guiding force for the agent in the environment to maximize the reward and is defined through a deterministic Bellman equation [29] as follows [13]:

$$\mathcal{Q}(S(t),\ Ac(t)) = \Big\{(1-n)\mathcal{Q}(S(t),\ Ac(t)) \\ + \eta(Re(t+1)) + \Upsilon^{\max}_{Ac(t+1)}\mathcal{Q}(S(t+1),\ Ac(t+1))\Big\} \quad (30)$$

where η is the learning rate; and Υ denotes the discount rate.

The Q-learning approach first initializes the Q-value to zero for all periods. The agent gains the state space $S(t)$ information from the environment (RSUs) at time t. According to the state, the agent selects an action $Ac(t)$ from the $\mathcal{Q}(S(t-1),\ Ac(t-1)\ )$ using an $\epsilon-$ greedy policy. Using this policy, the agent identifies the best action with $(1-\epsilon)$ probability rate and random action with ϵ probability rate. Further, a Q-value is updated in each iteration, and the mean reward is calculated afterwards.

B. Deep Q-Learning-Based Neural Network (DQNN) Approach

The Q-learning method is primarily used for a small set of state and action spaces. Since the Vehicular Fog Computing environment is dynamic in nature, the Q-learning approach is not suitable for finding the optimal reward function. Therefore, a Deep Neural Network is incorporated into the Q-learning approach with a parameter set θ to deal with the dynamic nature of tasks for optimal results. In DNN, θ is used to map the input state to action. In order to reduce the loss, a target layer is added to the primary neural network for network stabilization. The loss function of this neural network is expressed in Eq. (31) [31]:

$$l(\theta) = \left(O_{DQNN} - \mathcal{Q}(S(t),\ Ac(t))\right)^2 \quad (31)$$

where O_{DQNN} denotes the output of the DQNN and is expressed in Eq. (32):

$$O_{DQNN} = Re(t+1) + \max_{Ac(t+1)} \mathcal{Q}(S(t+1),\ Ac(t+1)) \tag{32}$$

For network stabilization, the transition table is updated with tuples, such as {*S*(*t*), *Ac*(*t*), *Re*(*t* + 1), *S*(*t* + 1)}. The DQNN works as follows: the parameters of the primary and target layers are initialized. The DQNN agent gains the state space information from all the RSUs, then identifies an action space using an ϵ-greedy policy. Next, the transition table is updated with the reward and state space data. Furthermore, the weights are updated, and the loss function specified in Eq. (31) is diminished using a gradient descent approach. The DQNN duplicates the parameters (θ) of a target layer from the primary network at the end of a specific phase.

C. FedDQL Approach

Federated learning (FL) is a cutting-edge concept that considers all the components to design a global model that does not need to share the original data. Herein, the core features of FL were incorporated with DQNN to develop the FedDQL global model, which has two primary advantages: (a) privacy preservation in terms of sharing confidential data, and (b) partial participation of the Fog servers that are energy-deficient. The Fog geographic region is scattered with clusters of Fog zones consisting of one or more RSUs and homogeneous and heterogeneous Fog servers. Each Fog zone has a DQNN agent to determine the offloading scheme with the DQNN approach.

In FL architecture, which is a distributed machine learning approach, there is a single model that is responsible for calculating and setting all the global metrics. This model is called the "unified model," and it acts as a central point of coordination for the distributed learning process. In the FedDQL model, the DQNN agent of each Fog zone downloads the global metrics from the unified model and trains their local model accordingly. After training, the metrics are updated and then forwarded to the unified model. Afterwards, the unified model aggregates all the local metrics and forms the global metrics. In this model, all the non-involving Fog zones merely download the unified model to keep an update.

This work considers *M* set of involving Fog zones of size *K* as $K = P|Z(f)|$, where *P* denotes the involvement factor, and *Z*(*f*) is the clusters of Fog zones. The involved Fog zones are identified according to their uplink transmission cost [22], which is calculated as follows:

$$Tr_m^{Up} = T_P^m \times \frac{l}{A} \tag{33}$$

where T_P^m denotes the power to transmit the local metrics by the m^{th} Fog zone of size *l*; and *A* is the rate taken to transmit by the m^{th} Fog zone of size *l* and is estimated as:

$$A = Bw \log_2\left(1 + \frac{T_p^m \times h}{\sigma^2 + \sum_{P=1,\ P\neq m}^{|Z(f)|} T_P^m \times h}\right) \tag{34}$$

where *Bw* is the channel's bandwidth; *h* is the channel gain; and $\sum_{P=1,\ P\neq m}^{|Z(f)|} T_P^m \times h$ is the surrounding noise caused by other Fog zones. Algorithm 1 shows the pseudocode for the FedDQL-based offloading strategy.

Algorithm 1 FedDQL-Based Offloading Strategy

Input: Cluster of Fog zones *Z*, Participation rate *P*; Set of vehicles *V*; Set of requests *T*;
Output: Optimal mapping of subtasks into Fog servers to have reduced energy consumption and computational latency

Start
1. Set $M = \varnothing,\ K = p|Z(f)|$;
2. For each Fog zone $m \in z(f)$ Estimate the transmission cost (Tr_m^{hp}) using Eq. (33);
3. Arrange the Fog zones in z(f) according to their transmission cost in ascending order;
4. Discover the initial k number of Fog zones from z(f) and add them into the set of M;
5. ***for*** f: 0 to F do
  5.1 For each Fog zone $m \in M$, update its parameter set O_m^f using the DQN approach
  5.2 Transfer the updated parameter cost to the centralized entity;
  5.3 Aggregation of all the local parameters takes place by the centralized entity as
$$\theta^{f+1} = \sum_{m\in M} \frac{D_m}{D} * \theta_m^f;$$
End

In step 1, the array of Fog zones *M* and the series of involving Fog zones are initialized. The cost of transmission by each Fog zone is estimated using Eq. (33). Afterwards, each Fog zone is sorted in ascending order based on their cost of transmission *m*. Step 4 identifies the first *k* number of Fog zones in FL, which involves a set of *F* rounds. In each epoch, each Fog zone updates their local parameter lists and broadcasts it to the unified model to form a global model in steps 5.1 and 5.2. In step 5.3, the unified model obtains a global model (θ^{f+1}) by performing aggregation at each round *f* of the local parameter (θ_m^f), which is expressed as [15]:

$$\theta^{f+1} = \sum_{m\in M} \frac{D_m}{D} * \theta_m^f \tag{35}$$

where D_m denotes the size of the local buffer of the m^{th} Fog zone; and *D* implies the total size of data estimated as $D = \sum_{m\in M} D_m$.

V. Experimental Assessment and Discussions

This section describes the extensive simulations that were performed on a real-world benchmark dataset for different conflicting scheduling parameters to assess the proposed strategy. The simulation parameters were first delineated, then a convergence analysis of the proposed technique was carried out. Afterwards, a comparative performance analysis was achieved for the considered scheduling parameters against some existing works.

A. Setting of the Simulation Environment

iFogSim over CloudSim is used as a simulator to instantiate a VFC simulation environment. This environment encompasses 10 Internet-enabled Vehicles, 3000 dynamic tasks, 296 Fog nodes/VMs, and 16 RSUs. There are multiple MFC servers in this architecture, and each MFC server is connected

TABLE II
SIMULATION PARAMETERS

Parameter	*Value*
Transmission Power	2±0.2 W [37]
Gain of the channel	144±1.44 dB
Noise power	1.5×10^{-8} W
Vehicle's computational capacity	0.3± 0.03 Gigacycle/s
Learning rate	0.01
MFC server's communication capacity	40 Mhz
MFC server's computational capacity	5 Gigacycle/s

TABLE III
TECHNICAL SPECIFICATIONS OF FOG NODES/VMS

Category	**Processing Cores**	**Speed, MIPS**	**Storage**	**Bandwidth, MIPS**
Large	8vCPUs	12500	1024 TB	102400
Medium	4vCPUs	6500	1024 GB	102400
Small	2vCPUs	3500	1024 GB	102400

to a single Road Side Unit (RSU), which is a device used in vehicular communication systems. Additionally, each of these servers has many cores, which are individual processing units within the server that can execute tasks simultaneously. The lengths of a series of tasks range from 0-15000 MIs with the size of each task ranging from a set of {30, 35, 40, 45, 50, 60} MB. Each task requires the computation resource requirements, which are arbitrarily allocated from a set of {0.6, 0.8, 1.0, 1.2, 1.4} Gigacycle/s. Each task is arbitrarily subdivided into 6-10 subtasks. The transmission power is dependent on the hardware used in the system [35]. In the IoT-Edge-Fog-Cloud environment, the transmission power of an IoT device can be dependent on various hardware components, such as the transceiver, amplifier, power supply, and processor. The efficiency of all these components impacts the transmission power of IoT devices. Table II lists the other simulated parameters used in the experiment, and the technical specifications of Fog nodes/VMs are presented in Table III.

A real-world dataset [34] was utilized to assess the performance. The dataset has an ETC matrix containing a number of tasks and machines with a corresponding computation ratio. This dataset has twelve instances in the form of u_x_tm with respect to uniformity in data (u), consistency (X) among data, and heterogeneity of tasks (t) and machines (m). The consistency among data was further classified into three groups: consistent (c), inconsistent (i), and semi-consistent (s). Likewise, the tasks and machines heterogeneity is categorized as high (h) or low (l). As a result, the twelve instances were formed by considering the aforementioned criteria. In addition, a variable number of tasks and machines was considered and formed into three groups: Group 1 (1000 $\times$ 96), Group 2 (2000 $\times$ 196) and Group 3 (3000 $\times$ 294) in the structure ($t \times m$).

To assess the performance of the devised FedDQL offloading algorithm, five baselines [23], [30], [31], [32], [33] were considered for analysis and comparison.

B. Convergence Analysis

To demonstrate the convergence analysis of FedDQL over other baseline algorithms, a convergence of the proposed method was analyzed with different learning rates, and then the convergence rate of other algorithms was analyzed.

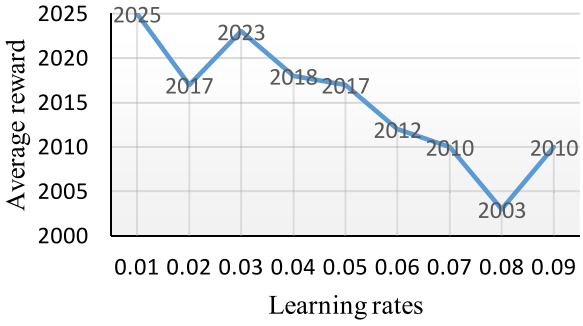

Fig. 3. Convergence analysis for different learning rates.

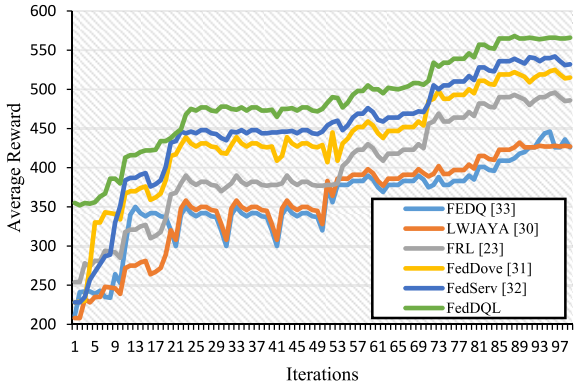

Fig. 4. Convergence analysis for different baselines.

The convergence of FedDQL was analyzed with disparate learning rates ranging from 0.01 to 0.09, as depicted in Figure 3. It is evident that the most suitable learning rate is 0.01 with a mean reward of 2025, while the worst learning rate is 0.08 with a mean reward of 2003. Therefore, 0.01 was considered in the simulation.

Figure 4 depicts the convergence rate of FedDQL and other compared algorithms. It can be observed that FedDQL converges faster due to the consideration of the dynamic behaviour of tasks and interdependencies among them. The optimal mapping of tasks into Fog servers or Cloud VMs results in fine-grained offloading.

C. Performance Analysis of Scheduling Metrics

Here, the performance of the FedDQL is analyzed and compared for different scheduling metrics, such as service time (ms), average utilization rate (%), mean energy consumption (KJ), and mean latency rate (ms). The proposed FedDQL is compared with other offloading strategies [23], [30], [31], [32], [33] to gauge its efficacy.

1) Performance Analysis of Service Rate: For latency-sensitive applications, the user's request must be serviced in a deterministic time and, hence, plays an indispensable part in this computing paradigm. Figure 5 shows a comparative analysis of the obtained results for service rate in three different sets (*tasks* $\times$ *machines*). The dew-enabled architecture and Deep Q-learning approach helped minimise the proposed framework's service time. Comparatively, the other approaches

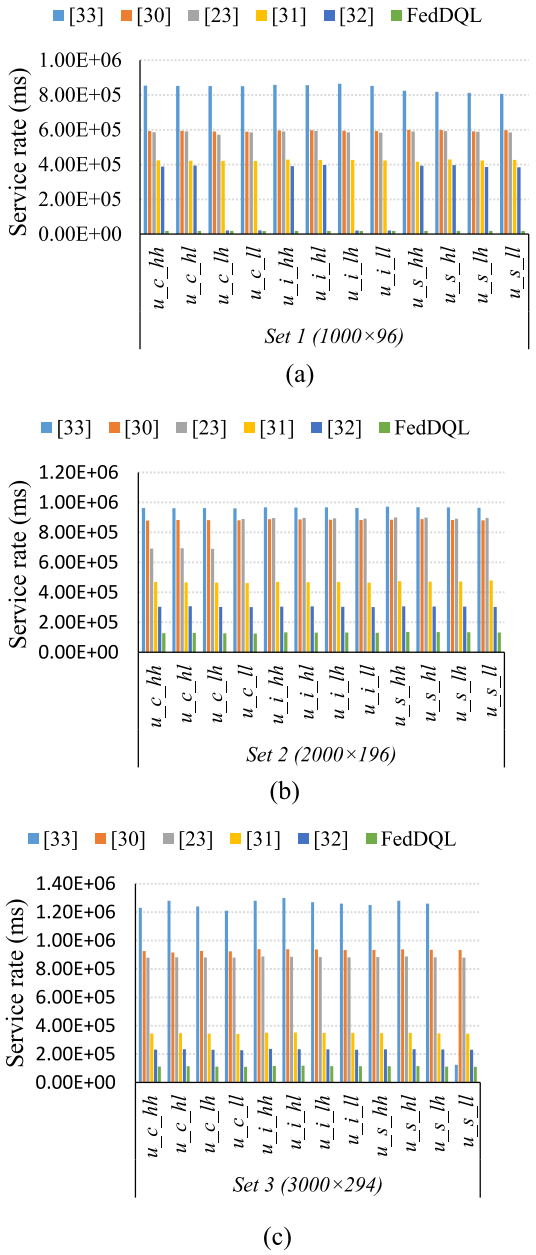

Fig. 5. QoS analysis of service rate (ms) for different sets: (a) 1000 × 96, (b) 2000 × 196, and (c) 3000 × 294.

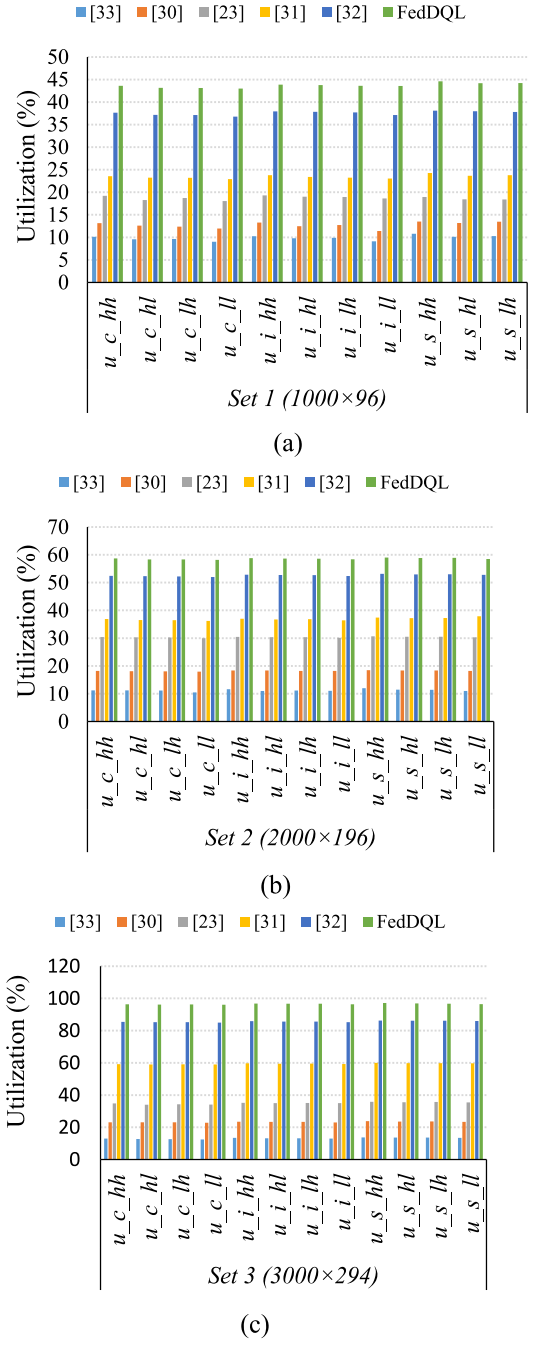

Fig. 6. QoS analysis of average utilization (%) for different sets: (a) 1000 × 96, (b) 2000 × 196, and (c) 3000 × 294.

do not consider the decomposition of tasks and classification of the target layers for offloading. Identifying different offloading layers has expedited the execution process and, thus, resulted in reduced service time for each user's request. The FedDQL outperforms the other models [23], [30], [31], [32], [33] with an improvement of 49%, 23.2%, 22.6%, 14.7% and 3.23% on average.

2) Performance Analysis of Average Utilization: Resource utilization is a significant part of offloading requests for resources. This factor contributes an equal amount with load balancing in meliorating the performance of any offloading algorithm. The use of the Deep Q-learning network helps to predict the load and assign the requests according to each Fog server, improving the utilization of all the underlying resources considerably.

Figure 6 depicts a comparative analysis of the obtained results for the average degree of utilization for three different sets. Nevertheless, all other approaches do not consider load balancing. The FedDQL outperforms the other models [23], [30], [31], [32], [33] with an improvement of 51%, 36.8%, 32.4%, 27.7% and 9.33% on average for three sets.

3) Performance Analysis of Latency Rate: Latency, which is caused by the high computational time or delay in transmission and propagation, is a pivotal element for latency-sensitive applications in computing paradigms. Due to the implementation of Fog servers between the Internet-enabled Vehicles and Cloud layer, the latency rate has been notably reduced. Moreover, classifying tasks and determining the offloading

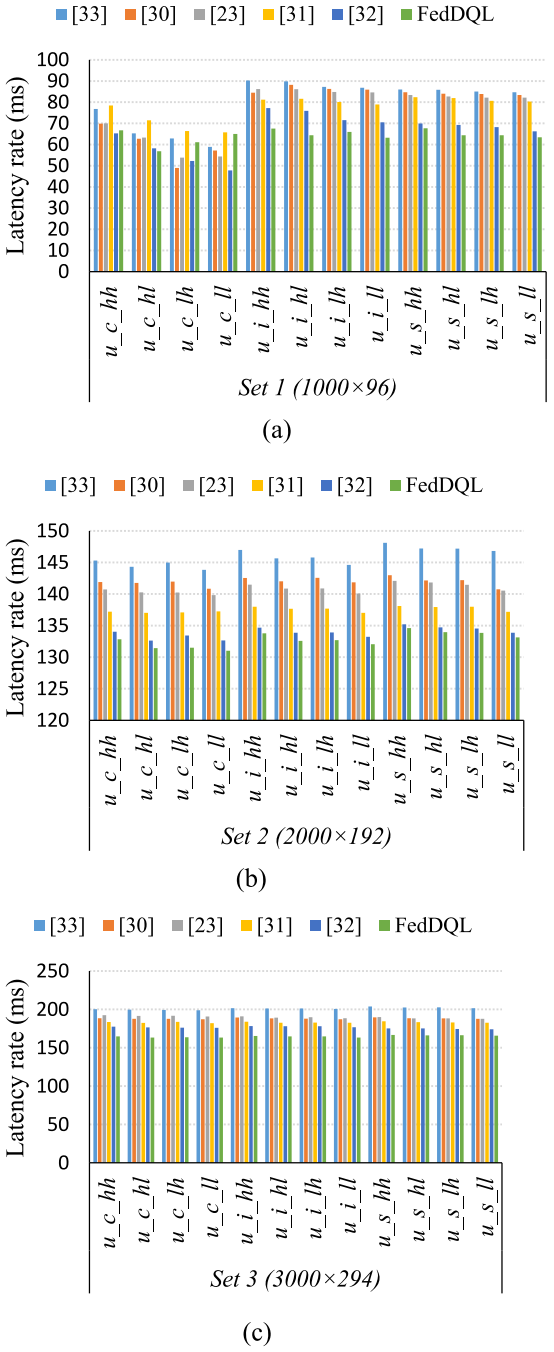

Fig. 7. QoS analysis of latency rate (ms) for different sets: (a) 1000 × 96, (b) 2000 × 196, and (c) 3000 × 294.

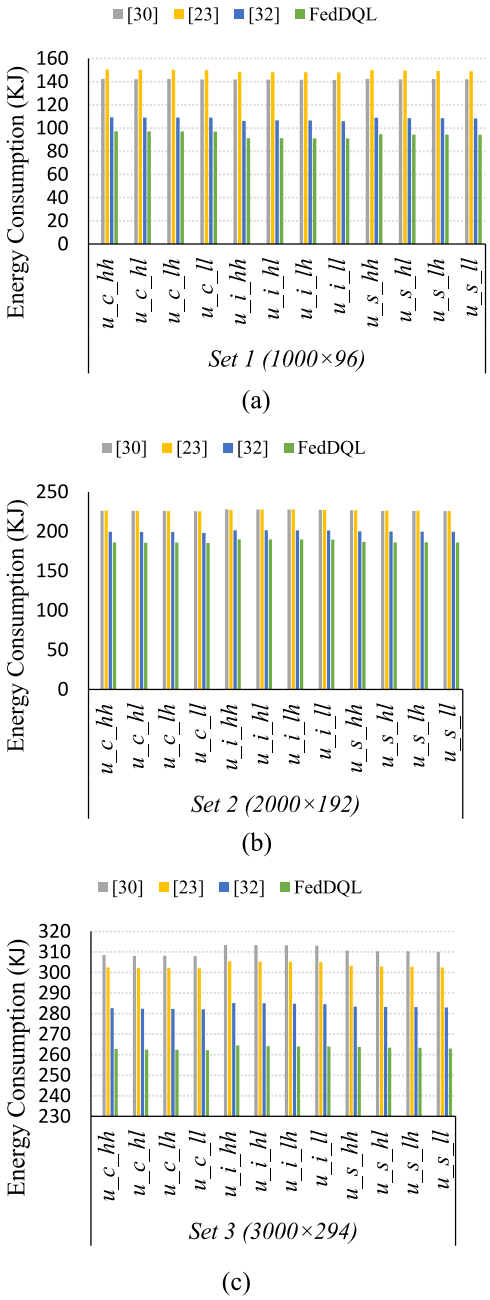

Fig. 8. QoS analysis of average energy consumption (KJ) for different sets: (a) 1000 × 96, (b) 2000 × 196, and (c) 3000 × 294.

layers also minimise the latency across computing layers. The latency rate holds significance when there is an increasing number of tasks in a dynamic environment. Figure 7 presents a comparative analysis of the obtained results for latency rate. It is obvious that the suggested method presents a remarkable improvement over the other models for different specifications. The FedDQL outperforms the other models [23], [30], [31], [32], [33] with an improvement of 15%, 9.4%, 9.9%, 14.7% and 2.33% on average for three sets. As the number of requests increase, the proposed method shows notable results over [32]. For a smaller set of requests, both of these (proposed and [32]) perform approximately.

4) Performance Analysis of Average Energy Consumption: The degree of energy consumption is a challenging factor for any datacentre. The consumption of energy depends on various factors, such as the specifications of the Fog servers/datacentre, computational capabilities, resource-constrained, size of the tasks, etc. In our approach, the energy consumption is evaluated for processing and storing each task on a resource, transmitting and acknowledging the requests from/to the computing layers. Efficient utilization of resources could minimize the degree of utilization. Figure 8 presents the QoS analysis of the degree of energy consumption for an increasing number of tasks for the proposed method. It is apparent that the suggested method reduces energy consumption more efficiently than the other methods with different degrees of

computational specifications. The FedDQL outperforms the other models [23], [30], [32] with an improvement of 39.9%, 34.7% and 21.6% on average for three sets.

For service rate, average utilization, latency and energy consumption, the effectiveness of the proposed FedDQL model has been compared to [23], [30], [31], [32], [33] and the improvement compared to [33] is 49%, 51% and 15%, the improvement of FedDQL compared to [30] is 23.2%, 36.8%, 9.4% and 39.9%, the improvement of FedDQL compared to [23] is 22.6%, 32.4%, 9.9% and 34.7%, the improvement of FedDQL compared to [31] is 14.7%, 27.7% and 14.7%, and the improvement of FedDQL compared to [32] is 3.23%, 9.33%, 2.33% and 21.6%, respectively.

VI. Conclusion and Future Study

This paper proposes a noble offloading method for compute-intensive and latency-sensitive dependency-aware tasks in the Dew-enabled collaborative computing framework. The dependency-aware tasks are depicted through a DAG, where the interdependencies among tasks are also modelled. Moreover, a Federated learning-supported Deep Q-learning (FedDQL)-based offloading strategy has been designed for the optimal assignment of tasks to machines. Next, Fuzzy logic is implemented to determine the target offloading layers to prevent starvation and aging. The simulation results showcase the efficacy of FedDQL over alternative offloading algorithms based on several performance metrics with different specifications. Tasks and machine heterogeneity are taken into account to appraise the effectiveness of the proposed method. The proposed FedDQL method outperforms others [23], [30], [31], [32], [33] with an average improvement of 49%, 34.3%, 29.2%, 16.2% and 8.21%, respectively.

A potential direction for future study are tasks offloading and data sharing while a vehicle is in motion. Also, data migration between MFC servers, and migration across computing layers in reverse could be a future scope.

References

[1] A. Boukerchea and R. E. De Grande, "Vehicular cloud computing: Architectures, applications, and mobility," *Comput. Netw.*, vol. 135, pp. 171–189, Apr. 2018.

[2] M. H. Eiza, Q. Ni, and Q. Shi, "Secure and privacy-aware cloud-assisted video reporting service in 5G-enabled vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7868–7881, Oct. 2016.

[3] A. M. A. Hamdi, F. K. Hussain, and O. K. Hussain, "Task offloading in vehicular fog computing: State-of-the-art and open issues," *Future Gener. Comput. Syst.*, vol. 133, pp. 201–212, Aug. 2022. [Online]. Available: https://doi.org/10.1016/j.future.2022.03.019

[4] S.-S. Lee and S. Lee, "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10450–10464, Oct. 2020, doi: 10.1109/JIOT.2020.2996213.

[5] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

[6] W. S. Atoui, W. Ajib, and M. Boukadoum, "Offline and online scheduling algorithms for energy harvesting RSUs in VANETs," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6370–6382, Jul. 2018.

[7] H. A. Khattak, S. U. Islam, I. U. Din, and M. Guizani, "Integrating fog computing with VANETs: A consumer perspective," *IEEE Commun. Standards Mag.*, vol. 3, no. 1, pp. 19–25, Mar. 2019.

[8] K. Kai, W. Cong, and L. Tao, "Fog computing for vehicular ad-hoc networks: Paradigms, scenarios, and issues," *J. China Univ. Posts Telecommun.*, vol. 23, no. 2, pp. 56–96, 2016. [Online]. Available: https://doi.org/10.1016/S1005-8885(16)60021-3

[9] L. Yao, X. Xu, M. Bilal, and H. Wang, "Dynamic edge computation offloading for Internet of Vehicles with deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 6, 2022, doi: 10.1109/TITS.2022.3178759.

[10] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 3448–3459, Sep. 2021.

[11] Z. Ning, P. Dong, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, 2019, doi: 10.1145/3317572.

[12] U. Maan and Y. Chaba, "Deep Q-network based fog node offloading strategy for 5G vehicular adhoc network," *Ad Hoc Netw.*, vol. 120, Sep. 2021, Art. no. 102565. [Online]. Available: https://doi.org/10.1016/j.adhoc.2021.102565

[13] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019, doi: 10.1109/TVT.2019.2935450.

[14] 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 Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2252–2261, Apr. 2021.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[16] C. Tang, X. Wei, C. Zhu, Y. Wang, and W. Jia, "Mobile vehicles as fog nodes for latency optimization in smart cities," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9364–9375, Sep. 2020, doi: 10.1109/TVT.2020.2970763.

[17] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 87–93, Feb. 2019, doi: 10.1109/MWC.2019.1700441.

[18] A. Thakur and R. Malekian, "Fog computing for detecting vehicular congestion, an Internet of Vehicles based approach: A review," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 2, pp. 8–16, Mar. 2019, doi: 10.1109/MITS.2019.2903551.

[19] Y. Li, H. Li, G. Xu, T. Xiang, and R. Lu, "Practical privacy-preserving federated learning in vehicular fog computing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4692–4705, May 2022.

[20] O.-K. Shahryari, H. Pedram, V. Khajehvand, and M. D. TakhtFooladi, "Energy-efficient and delay-guaranteed computation offloading for fog-based IoT networks," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107511, doi: 10.1016/j.comnet.2020.107511.

[21] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019, doi: 10.1109/TVT.2019.2894851.

[22] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020, doi: 10.1109/TVT.2020.3041929.

[23] F. H. Rahman, S. H. S. Newaz, T.-W. Au, W. S. Suhaili, M. A. P. Mahmud, and G. M. Lee, "EnTruVe: ENergy and TRUst-aware virtual machine allocation in VEhicle fog computing for catering applications in 5G," *Future Gener. Comput. Syst.*, vol. 126, pp. 196–210, Jan. 2022.

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

[25] A. R. Hameed, S. ul Islam, I. Ahmad, and K. Munir, "Energy- and performance-aware load-balancing in vehicular fog computing," *Sustain. Comput. Inform. Syst.*, vol. 30, Jun. 2021, Art. no. 100454.

[26] B. Shabir, A. U. Rahman, A. W. Malik, R. Buyya, and M. A. Khan, "A federated multi-agent deep reinforcement learning for vehicular fog computing," *J. Supercomput.*, vol. 79, pp. 6141–6167, Oct. 2022. [Online]. Available: https://doi.org/10.1007/s11227-022-04911-8

[27] X. Xu, Z. Fang, L. Qi, W. Dou, Q. He, and Y. Duan, "A deep reinforcement learning-based distributed service of loading method for edge computing empowered Internet of Vehicles," *Chin. J. Comput.*, vol. 44, no. 12, pp. 2382–2405, 2021.

[28] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
[29] T. Hester et al., "Deep Q-learning from demonstrations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 3223–3230.
[30] C. Chakraborty, K. Mishra, S. K. Majhi, and H. K. Bhuyan, "Intelligent latency-aware tasks prioritization and offloading strategy in distributed fog-cloud of things," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 2099–2106, Feb. 2023, doi: 10.1109/TII.2022.3173899.
[31] V. Sethi and S. Pal, "FedDOVe: A federated deep Q-learning-based offloading for vehicular fog computing," *Future Gener. Comput. Syst.*, vol. 141, pp. 96–105, Apr. 2023.
[32] M. Tiwari, I. Maity, and S. Misra, "FedServ: Federated task service in fog-enabled Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20943–20952, Nov. 2022, doi: 10.1109/TITS.2022.3186401.
[33] D. B. Son, V. T. An, T. T. Hai, B. M. Nguyen, N. P. Le, and H. T. T. Binh, "Fuzzy deep Q-learning task offloading in delay constrained vehicular fog computing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–8, doi: 10.1109/IJCNN52387.2021.9533615.
[34] T. D. Braun et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 61, no. 6, pp. 810–837, 2001. [Online]. Available: https://doi.org/10.1006/jpdc.2000.1714
[35] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.

Kaushik Mishra (Member, IEEE) received the Ph.D. degree from the Veer Surendra Sai University of Technology, Burla, in 2021. He is currently working as an Assistant Professor with the Department of Computer Science and Engineering, Gandhi Institute of Technology and Management University, Visakhapatnam, India. He has publications in various journals of repute and conference proceedings. His research area includes cloud computing and fog computing. He has received two best paper awards in two conferences held at NIT, Agartala, India, and ITER, Bhubaneswar, India, in 2020.

Goluguri N. V. Rajareddy is currently pursuing the Ph.D. degree with NIT Silchar and also working as an Assistant Professor with the Department of Computer Science and Engineering, School of Technology, Gandhi Institute of Technology and Management University (Deemed to be University), Visakhapatnam, India. He has published five research papers in various reputed international journals. He has eight years of full-time teaching experience and his areas of interest are image processing and deep learning. He is a member of IAENG, UACEE, and IFERP.

Umashankar Ghugar (Member, IEEE) received the Doctoral degree (Full-Time) from Berhampur University, Odisha, in 2021. He is currently working as an Assistant Professor with the School of Technology, Department of CSE, GITAM University, Visakhapatnam. He has published 14 articles, including journals, book chapters, and conferences in international publishers. His research interests are in computer networks and network security in WSN. He is a Reviewer of IEEE ACCESS, IEEE TRANSACTION ON EDUCATION, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Security and Privacy* (Wiley), *International Journal of Communication Systems* (Wiley), *International Journal of Distributed Sensor Networks* (Hindawi), *International Journal of Knowledge Discovery in Bioinformatics* (IGI Global), and *International Journal of Information Security and Privacy* (IGI Global) and a member of IACSIT, CSTA, and IRED.

Gurpreet Singh Chhabra received the Ph.D. degree from Department of Computer Science and Engineering. He is currently working as an Assistant Professor with the Computer Science and Engineering Department, GITAM School of Technology, GITAM (Deemed to be University), Visakhapatnam. He has 15 years of teaching experience. His qualifications are fortified with a great deal of creativity and problem-solving skills. He has credit for many international papers, patents, book chapters, and books. His research interests in deep learning, machine learning, data science, and fog computing. He is a Life Member of the ISTE and IAENG.

Amir H. Gandomi (Senior Member, IEEE) is a Professor of Data Science and an ARC DECRA Fellow with the Faculty of Engineering and Information Technology, University of Technology Sydney. He is also affiliated with Óbuda University, Budapest, as a Distinguished Professor. Prior to joining UTS, he was an Assistant Professor with the Stevens Institute of Technology, USA, and a Distinguished Research Fellow with BEACON center, Michigan State University, USA. He has published over three hundred journal papers and 12 books which collectively have been cited 4000+ times (H-index = 91). He has been named as one of the most influential scientific minds and received the Highly Cited Researcher Award (top 1% publications and 0.1% researchers) from Web of Science for six consecutive years from 2017 to 2022. In the recent most impactful researcher list, done by Stanford University and released by Elsevier, and he is ranked among the top 1,000 researchers (top 0.01%) and top 50 researchers in AI and Image Processing subfield in 2021. He also ranked 17th in GP bibliography among more than 15 000 researchers. His research interests are global optimization and (big) data analytics using machine learning and evolutionary computations in particular. He has received multiple prestigious awards for his research excellence and impact, such as the 2022 Walter L. Huber Prize and the Highest-Level Mid-Career Research Award in all areas of civil engineering. He has served as an Associate Editor, an Editor, and the Guest Editor for several prestigious journals, such as an Associate Editor for IEEE NETWORKS and IEEE INTERNET OF THINGS JOURNAL. He is active in delivering keynotes and invited talks.