Computer Modeling in Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.062642

ARTICLE

A Privacy-Preserving Graph Neural Network Framework with Attention Mechanism for Computational Offloading in the Internet of Vehicles

Aishwarya Rajasekar* **and Vetriselvi Vetrian**

Department of Computer Science and Engineering, College of Engineering Guindy, Anna University, Chennai, 600025, India

*Corresponding Author: Aishwarya Rajasekar. Email: 22244197106@annauniv.edu

Received: 23 December 2024; Accepted: 27 February 2025; Published: 11 April 2025

ABSTRACT: The integration of technologies like artificial intelligence, 6G, and vehicular *ad-hoc* networks holds great potential to meet the communication demands of the Internet of Vehicles and drive the advancement of vehicle applications. However, these advancements also generate a surge in data processing requirements, necessitating the offloading of vehicular tasks to edge servers due to the limited computational capacity of vehicles. Despite recent advancements, the robustness and scalability of the existing approaches with respect to the number of vehicles and edge servers and their resources, as well as privacy, remain a concern. In this paper, a lightweight offloading strategy that leverages ubiquitous connectivity through the Space Air Ground Integrated Vehicular Network architecture while ensuring privacy preservation is proposed. The Internet of Vehicles (IoV) environment is first modeled as a graph, with vehicles and base stations as nodes, and their communication links as edges. Secondly, vehicular applications are offloaded to suitable servers based on latency using an attention-based heterogeneous graph neural network (HetGNN) algorithm. Subsequently, a differential privacy stochastic gradient descent training mechanism is employed for privacy-preserving of vehicles and offloading inference. Finally, the simulation results demonstrated that the proposed HetGNN method shows good performance with 0.321 s of inference time, which is 42.68%, 63.93%, 30.22%, and 76.04% less than baseline methods such as Deep Deterministic Policy Gradient, Deep Q Learning, Deep Neural Network, and Genetic Algorithm, respectively.

KEYWORDS: Internet of vehicles; vehicular *ad-hoc* networks (VANET); multiaccess edge computing; task offloading; graph neural networks; differential privacy

1 Introduction

With the emergence of 6G, intelligent transportation systems (ITSs) and Vehicular *Ad-hoc* Networks (VANET) have made the Internet of Vehicles (IoV) a rapidly growing area [1]. The IoV facilitates the connection of vehicles, city infrastructure, and pedestrians to the Internet, enabling the exchange of information. It allows the sharing of data among vehicles, i.e., Vehicle-to-Vehicle (V2V), humans, i.e., Vehicle-to-Pedestrian (V2P), infrastructure, i.e., Vehicle-to-Infrastructure (V2I), and cloud/edge platforms, i.e., Vehicle-to-Network (V2N) establishing essential Vehicle-to-Everything (V2X) links [2]. The use of fleet management, in-vehicle entertainment, internet access, roadside assistance, vehicle diagnostics, navigation, advanced driver assistance services, and traffic-efficient applications [3] that reduce transit time and avoid idle times are gradually increasing in connected vehicles. As a result, smart vehicles are generating vast amounts of data, placing increased demands on the IoV in terms of computational capacity, energy consumption, and processing latency. Limited processing capacity on vehicles cannot process all the data. However,

processing IoV-generated data in the cloud presents challenges such as high bandwidth consumption and network bottlenecks, resulting in significant delays. Latency-sensitive applications such as accident warnings, safety alert systems, and collision avoidance systems, cannot tolerate latency, thus necessitating nearby processing of data.

Multiaccess Edge Computing (MEC) and fog computing have garnered attention for processing applications at the edge in the IoV [4]. MEC integrated with a vehicular network forms a vehicular edge computing (VEC) paradigm. MEC servers are positioned at the edge of the vehicular network, i.e., Road Side Units (RSUs), base stations (BSs), and unmanned aerial vehicles (UAVs), to support vehicular applications. A space air-ground integrated network can be combined with a VANET to support ubiquitous coverage and intelligence for emerging applications in a flexible, low-latency manner. In the 6G-enabled IoV, the Space Air Ground Integrated Vehicular Network (SAGIVN) plays a significant role in enabling connected intelligence for its applications by providing communication in rural areas, and emergency or disaster environments with the help of ground, aerial, and satellite communication [5].

The heterogeneous, dynamic, real-time IoV environment and resources as well as energy-constrained devices in the IoV make computation offloading challenging. Additionally, the tasks are highly heterogeneous depending on the priorities, deadlines, arrival times, and resource requirements. Hence, an efficient computational offloading algorithm can greatly enhance the quality of service (QoS) of vehicular applications by reducing delays. Various studies have explored diverse techniques, such as metaheuristic algorithms [6–8], game theoretic solutions [9,10], and deep reinforcement learning (DRL) [11,12], to address resource allocation and computation offloading problems.

Despite significant advancements in task-offloading methods [9–13], certain challenges persist. A primary concern is the complexity of the existing algorithms in terms of convergence time and inference efficiency. Ensuring robustness with varying numbers of vehicles, servers, and computational resources also presents difficulties. Additionally, a privacy-preserving mechanism is essential, as the offloading framework handles diverse data from vehicular devices to infrastructure. To address these challenges, this work introduces a privacy-preserving graph attention-based framework that enhances scalability and data security.

Task offloading decisions have to be made using a lightweight algorithm with minimal inference time to reduce the offloading failure rate of vehicular applications on resource-constrained devices. Thus, instead of deep reinforcement learning algorithms, simple graph neural networks (GNN) shall be incorporated for task offloading. Inspired by the recent successes of the application of deep learning in many domains [14–17], recent research has shown a growing interest in incorporating graph neural networks (GNN) into vehicular environments [18–20] as well. The components in the IoV environment are depicted as nodes in a graph, with their relationships represented as edges. The heterogeneous graph is best suited for the representation of a vehicular network and is processed using GNN.

A secure IoV environment is crucial because compromising the security of the IoV may lead to serious fatal effects for humans [21]. The edge server component may be curious and may attempt to infer the vehicle data and offload inference information from the trained model. It may then mislead the vehicles to offload the computation to the malicious RSUs by deducing the learned algorithm and crafting the specific inputs tailored to deceive the vehicles. Thus, privacy-preserving techniques such as encryption, and differential privacy [22,23] should be incorporated to enhance the protection of vehicular data and offloading inference in the proposed HetGNN algorithm.

In this paper, a novel privacy-preserving unsupervised HetGNN-based deep learning algorithm is introduced for secure computation offloading in the IoV environment. Compared to the existing works, our proposed work represents the IoV environment as a graph, and with a secure heterogenous GNN, it makes

the optimal offloading decision by minimizing the overall execution delay of the tasks with minimal inference time. The GNN can provide topology-independent feature extraction. The structural characteristics of networks are exploited by GNNs by propagating node features across edges [24]. The use of the GNN can generalize well for unseen topologies without retraining, making it suitable for IoV architecture. On the other hand, this allows for making offloading decisions during all kinds of hours of the day with varying numbers of vehicles within the particular BS coverage area, supporting scalability. To the best of our knowledge, this is the very first attempt at applying Graph Neural Network for task-offloading decisions in the IoV. Additionally, to safeguard vehicle privacy and prevent offloading inference attacks, a Differential Privacy Stochastic Gradient Descent (DPSGD) based training mechanism is considered for the HetGNN. The key contributions of the proposed work are outlined as follows:

1. Instead of relying on complex algorithms like DRL, a simple GNN-based framework is incorporated to make the task-offloading decision. This approach offers a comprehensive representation of the SAGIVN architecture by modeling it as a heterogeneous graph. By leveraging the advantages of the graphical representation of vehicular networks, GNN can effectively learn the features of heterogeneous nodes and capture the link among them for precise task-offloading decisions.
2. We propose a secure, unsupervised HetGNN task offloading algorithm based on a graph attention network. The GNN with an attention mechanism is employed for task offloading and is trained using an unsupervised method by focusing on minimizing the overall execution delay of the task, thereby ensuring the QoS of vehicular applications. It employs a DPSGD training mechanism to preserve the privacy of individual vehicles and secure the overall IoV system. This mitigates the effectiveness of adversaries to gain unauthorized access to sensitive information about the IoV system.
3. We trained the framework with synthetically generated graph data with low and high numbers of vehicles and edge servers and its computational complexities, along with a privacy budget, to stimulate complex environments, enhance robustness and scalability, and preserve privacy. This ensures the model can effectively handle varying traffic conditions and resource availability, improving its generalization capabilities.
4. Extensive experimental analysis is conducted on a synthetically created dataset to demonstrate the proposed HetGNN algorithm's effectiveness. Experimental findings show that the proposed algorithm exhibits faster convergence, lower inference time delays, and a low task failure rate compared to existing baseline methods.

The rest of the article is organized as follows: Section 2 reviews the literature. Section 3 provides the SAGIVN architecture, the system model, and the problem statement. The proposed secure unsupervised HetGNN task offloading algorithm is detailed in Section 4. Experimentation and results analysis are discussed in Section 5, and the conclusions are presented in Section 6.

2 Related Works

This section presents literature describing computation offloading in the IoV environment, the integration of GNNs for the IoV system model, and secure computation offloading.

2.1 Task Offloading in the IoV

Cost, time, latency, energy, load balancing, and resource utilization are the important parameters considered in the literature for task offloading. Deng et al. [25] examined the application of an alternate convex search algorithm to find the best offloading strategy with low computational complexity for minimizing latency and energy consumption costs. Mirza et al. [26] proposed a contact, mobility, and computation load-aware task offloading scheme for heterogeneous VECs by considering the 5G-NR-V2X standard along

with mmWave communications to lower the system cost and demonstrated a significant improvement in the task turnover ratio. Alseid et al. [6] adopted multihop task offloading using the best candidate vehicle selection mechanism to address the MEC server coverage area limitation, using the modified sparrow search algorithm.

Zhang [11] studied the use of an analytical approach for the offloading strategy of multiuser MEC scenarios and proposed the Brute force-based Deep Deterministic Policy Gradient (DDPG) algorithm, where the IoV system's structure is split into various groups to increase the offloading performance. Sun et al. [27] focused on task offloading in the point of interest-based Vehicular Fog Computing (VFC) scenario and proposed delay-aware and vehicle resource-aware task offloading algorithms that maximize the task completion rate. Wu et al. [13] researched the scheme of task offloading in a VFC-assisted platoon system by considering the heterogeneous resources of vehicles and the random arrival and departure times of tasks and vehicle, using semi-Markov decision process.

Fan et al. [28] and Wan et al. [7], considering the joint optimization of resource allocation and task offloading, proposed an algorithm based on generalized bender decomposition and reformulation linearization methods and an improved biogeography-based optimization to reduce energy consumption and latency. Alruwais et al. [8] applied a farmland fertility algorithm for resource scheduling to reduce the overall execution time of the tasks. However, most existing task offloading algorithms face challenges in ensuring convergence, and DRL-based approaches often require extensive training, demanding significant computational resources. Therefore, a lightweight and computationally efficient task offloading strategy is required.

2.2 GNN-Based Approaches in the IoV

Recently, the use of graph structures has progressed to diverse domains spanning wireless networks [29], traffic systems [20], and social networks [30]. GNN, an extension of deep learning, handles graph data by integrating the graph's structure and node embeddings through iterative message passing and aggregation [24]. Kumar et al. [31] introduced a fuzzy-based GNN to optimize network selection for IoV communication. Nazzal et al. [32] focused on predicting taxi service demand and supply by modeling the taxi system as a heterogeneous graph using Graph Neural Network and Long Short-Term Memory (GNN-LSTM) approach. Sant'Ana da Silva et al. [33] investigated the application of GNNs to support decision-making in collective intelligent transportation systems. They can generate synthetic data to train deep learning models effectively and can be used for accident prevention and traffic analysis in ITSs.

Chen et al. [34] constructed a knowledge graph-aware framework for air-ground collaboration offloading and content acquisition models in the VANET environment. Depending on the multi-temporal parametric knowledge relations, a dynamic orchestration algorithm is proposed for the edge service knowledge graph. Mehmood et al. [18] enabled a proactive platform in the IoV for traffic forecasting by integrating a GNN and a Gated Recurrent Unit (GRU) to explore the spatial and temporal features of multiclass traffic densities. The GNN has also been used for trajectory prediction [19] with an attention-based Graph Convolutional Network-Bidirectional Long Short-Term Memory (GCN-BiLSTM) algorithm and for traffic flow prediction [20] using a graph-weighted convolution network. These literatures ensure the applicability of GNN for the IoV environment to effectively capture the connectivity of the nodes leveraging the spatial relations. Thus, in the proposed work IoV environment is considered a heterogeneous graph and GNN has been used for making task-offloading decisions.

2.3 Secure Task Offloading in the IoV

Salami et al. [35] proposed a solution for secure offloading based on elliptic curve cryptography techniques. Li et al. [36] explored blockchain-enabled task offloading within the digital twin vehicular network. Blockchain is utilized to ensure secure and decentralized offloading transactions. The improved Cuckoo algorithm is employed for task offloading considering latency and energy consumption. Mao et al. [37] proposed the task offloading system for the IoV centered around a trusted RSU. Each RSU assesses trust values based on real-time data, updating them in local databases and obtaining global trust values from a central trust authority. Nodes exceeding a specific trust threshold are labeled as malicious. Task offloading occurs only to trusted RSUs within coverage, with tasks delayed until a trusted RSU becomes available if necessary.

Location privacy in the IoV is critical because it may cause life-threatening issues. The current location of vehicles is inferred by attackers based on offloading behavior as it has a certain regularity. Gao et al. [38] focused on location privacy-based task offloading. The privacy loss model is designed along with the optimization objective and DRL is used to solve the offloading problem with location privacy. Lakhan et al. [22] presented the fully homomorphic-enabled secure task offloading for mobility-enabled vehicle applications. The task computation is performed on the encrypted data, thus ensuring secure data transmission.

The aforementioned studies demonstrate diverse attempts to address challenges such as task offloading, secure task offloading, and applying GNNs for a range of purposes within the IoV framework. However, there remains a clear demand for secure, lightweight and computationally efficient task offloading strategy. This requirement has made the authors to propose a solution for secure task offloading, based on GNN and differential privacy techniques. In this approach computational offloading along with privacy preservation and scalability of the IoV environment are comprehensively considered. This paper introduces a pioneering approach, presenting a novel secure unsupervised HetGNN task offloading algorithm. This innovative model is mounted on the BSs to make informed offloading decisions.

3 System Architecture

The Space Air Ground Integrated Vehicular Network (SAGIVN) architecture is introduced first. Subsequently, the relevant models and problem formulations are defined.

3.1 Space Air Ground Integrated Vehicular Network Architecture

The Space Air Ground Integrated Vehicular Network (SAGIVN) architecture is made possible with the 6G communication technology and it supports seamless connectivity. It comprises three layers: the ground layer, the air layer, and the space layer. The multitier architecture of SAGIVN is illustrated in Fig. 1. Ground layer: The ground layer contains vehicles, BSs, RSUs, and MEC servers. Ground networks such as BSs and RSUs offer services to high vehicle density areas. Air layer: The air network is composed of UAVs, and High Altitude Platform Stations (HAPs) fitted with edge servers and is mobile, enabling dynamic deployment. Space layer: The space layer is composed of commercial satellites that provide ubiquitous connectivity.

In practical scenarios, the vehicular applications are diverse and require different QoSs. To accommodate these services in both rural and urban areas, the advantages of the heterogeneous networking paradigm must be exploited. Aside from terrestrial networks, both aerial and satellite networks are being explored with the advancement of 6G technology. The SAGIVN aims to provide robust and ubiquitous connectivity for vehicles. Ground-based networks such as BSs and RSUs support high data access rates in urban areas. Air-based networks such as UAVs and HAPs provide improvements in high-service demand areas and

emergency or disaster situations. Space-based networks, such as satellites, provide connectivity to rural areas and challenging environments.

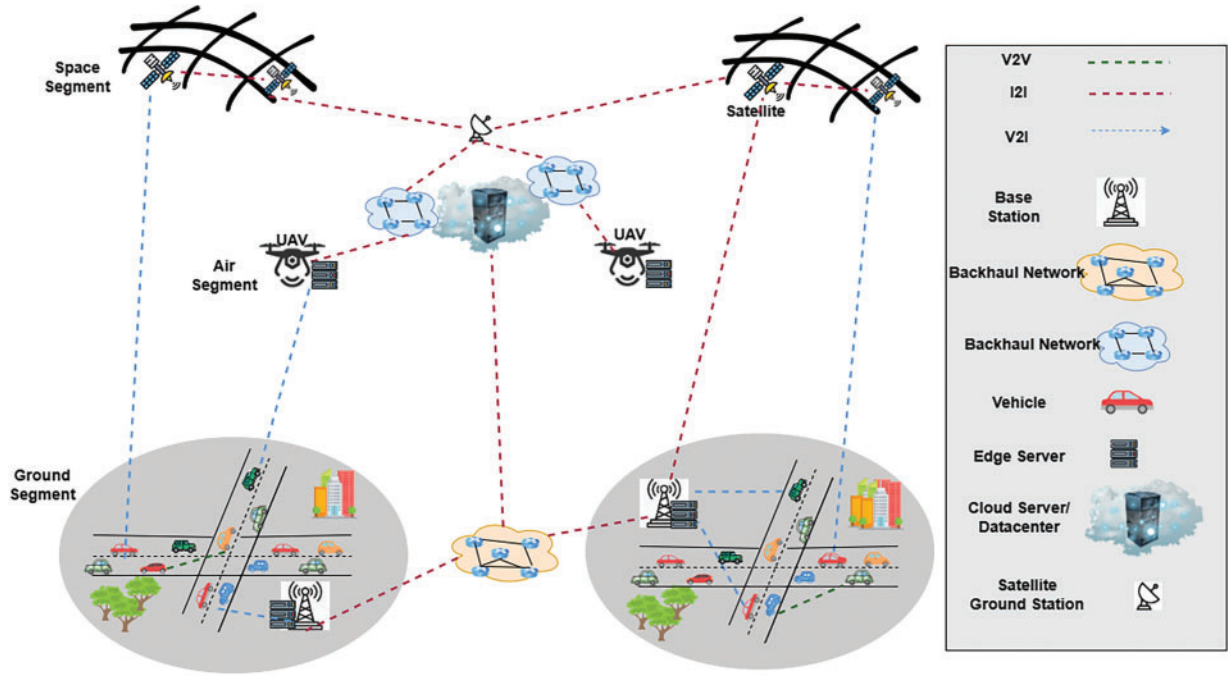

Figure 1: Multitier architecture of space air ground integrated vehicular network

Thus, SAGIVN provides an integrated communication network, seamless connectivity, low latency, improved throughput, and network congestion alleviation for the IoV environment to offer improved driving experience and intelligent transportation services. For task offloading, this ubiquitous connected architecture enables the dynamic allocation of tasks from vehicles to various base stations depending on the QoS requirements and network coverage by ensuring optimized resource usage and reduced delays.

Ground BSs are equipped with edge servers to provide edge computing while air layer UAVs and HAPs, also provide edge services in insufficient ground coverage areas. Space layer satellites act as relays in the areas where the ground or air base stations are affected. Thus, the integration of ground, air, and space-based BSs overcome coverage limitations ensuring seamless communication even during disaster scenarios. This expanded coverage enhances reliability, and network availability, improving the overall performance of task offloading [5]. With this architecture, the system model and problem formulation for task offloading are outlined in the following section.

3.2 System Model

A novel secure unsupervised HetGNN task offloading algorithm is proposed by modeling SAGIVN as a heterogeneous graph. It leverages task requirements, and resource availability to minimize the overall execution delay of tasks, thereby ensuring the QoS of vehicular applications. The differentially private stochastic gradient descent (DPSGD) training mechanism is incorporated into the proposed secure unsupervised HetGNN task offloading algorithm to mitigate the effectiveness of adversaries in preserving the privacy of individual vehicles and offloading inference attack protection.

The SAGIVN is modelled as a heterogeneous graph, where the nodes correspond to vehicles, (RSUs, BSs, UAVs)-associated servers, cloud servers, and satellites. Edges denote communication links with transmission rates capturing connection quality. This graph-based representation encapsulates the unique characteristics of different SAGIVN components, such as the task requirements of vehicles, the computational resources of servers and transmission rates between nodes. The heterogeneous graph structure guides the Graph Attention Network in task-offloading decisions by analyzing node and edge characteristics. HetGNN leverages the connectivity of SAGIVN by dynamically offloading vehicular tasks to nearby edge servers for low-latency processing, more distant servers when local resources are insufficient, or cloud servers for high-computation tasks. Furthermore, the interconnected nature of SAGIVN, which encompasses communication among vehicles, edge servers, and the cloud, can be represented in a heterogeneous graph to provide an interconnected network view. This heterogeneous graph representation enables the optimization of task offloading by minimizing the overall execution delay of vehicular tasks while maintaining the QoS. Thus, the HetGNN-based solution exploits SAGIVN's inherent connectivity, scalability, and resource diversity. The IoV system is modeled as follows.

A heterogeneously distributed SAGIVN environment consisting of $\boldsymbol{V}$ vehicles, $\boldsymbol{G}$ ground BSs, $\boldsymbol{U}$ air BSs, and a satellite (**ST**) is considered. Here RSUs, UAVs, and **ST** are represented as BSs. The vehicles interact with the BS within its coverage area. All BSs are connected to each other and connected to the cloud through the core network. The set of vehicles in the environment represented by $\mathbf{V} = \{\mathbf{1,2}, \ldots, \boldsymbol{V}\}$ is fully covered by either a ground BS RSUs $\mathbf{G} = \{\mathbf{1}, \ldots \boldsymbol{G}\}$, an air BS UAVs $\mathbf{U} = \{\mathbf{1}, \ldots \boldsymbol{U}\}$ or a satellite **ST**. Vehicles interacts with BSs through V2I links. The ground and air BSs are equipped with edge servers. Let $\mathbf{R} = \{\mathbf{G} \cup \mathbf{U} \cup \mathbf{CS}\}$ represents the set of servers in the RSUs, the UAVs, and the cloud **CS**. The vehicle has a diverse range of applications, including path planning, multimedia, security applications, etc., each of which may generate distinct tasks. Each vehicle communicates with BSs for task offloading purposes. The execution of tasks at different destinations has varying delays. The adversaries may compromise the privacy of vehicles as well as infer the offloading preferences. Thus, the task model, communication model, delay model, and adversary model for task offloading have been detailed below. Table 1 describes the various notations used.

Table 1: Notations and descriptions

Notation	*Description*
$\boldsymbol{X}(\boldsymbol{t})$	Task offloading indicator set
$\boldsymbol{x}_{\boldsymbol{vk}}(\boldsymbol{t})$	Processing of task $\boldsymbol{k}$ locally in vehicle $\boldsymbol{v}$
$\boldsymbol{x}_{\boldsymbol{mnk}}(\boldsymbol{t})$	Processing of task $\boldsymbol{k}$ in BS $\boldsymbol{n}$
$\boldsymbol{x}_{\boldsymbol{mmk}}(\boldsymbol{t})$	Processing of task $\boldsymbol{k}$ in BS $\boldsymbol{m}$
$\boldsymbol{x}_{\boldsymbol{mck}}(\boldsymbol{t})$	Processing of task $\boldsymbol{k}$ in cloud **CS**
$\boldsymbol{x}_{\boldsymbol{k}}(\boldsymbol{t})$	The offloading decision of task $\boldsymbol{k}$
$\boldsymbol{r}_{\boldsymbol{ij}}$	Uplink transmission rate

Task Model: In time slot t, the vehicles may generate task demands denoted by $\mathbf{TD} = \{\mathbf{1,2}, \ldots, \boldsymbol{TD}\}$. Each task $\mathbf{k}$ belongs to $\boldsymbol{TD}$ and can be represented as $\boldsymbol{k} = (\boldsymbol{d}_k, \boldsymbol{c}_k, \boldsymbol{l}_k)$. Task $\boldsymbol{k}$ is defined by three elements as follows: the task's data size $(\boldsymbol{d}_k)$, the task's computation requirements $(\boldsymbol{c}_k)$, and the latency constraint $(\boldsymbol{l}_k)$. The proposed scheme incorporates a GNN to make offloading decisions for tasks generated by vehicles. The task offloading indicator set $\boldsymbol{X}(\boldsymbol{t})$ denotes the range of potential destinations for offloading and is symbolized

as depicted in Eq. (1).

$$\boldsymbol{X}(\boldsymbol{t}) = \{\boldsymbol{x}_{\boldsymbol{vk}}(\boldsymbol{t})\}_{\substack{\boldsymbol{v}\in\boldsymbol{Vehicle},\\ \boldsymbol{k}\in\boldsymbol{TD}}} \cup \{\boldsymbol{x}_{\boldsymbol{mnk}}(\boldsymbol{t})\}_{\substack{\boldsymbol{m},\boldsymbol{n}\in\boldsymbol{BS},\\ \boldsymbol{k}\in\boldsymbol{TD}}} \cup \{\boldsymbol{x}_{\boldsymbol{mmk}}(\boldsymbol{t})\}_{\substack{\boldsymbol{m}\in\boldsymbol{BS},\\ \boldsymbol{k}\in\boldsymbol{TD}}} \cup \{\boldsymbol{x}_{\boldsymbol{mck}}(\boldsymbol{t})\}_{\substack{\boldsymbol{m}\in\boldsymbol{BS},\boldsymbol{c}\in\boldsymbol{CS},\\ \boldsymbol{k}\in\boldsymbol{TD}}} \tag{1}$$

where $\boldsymbol{x}_{\boldsymbol{vk}}(\boldsymbol{t}) = \mathbf{1}$ indicates that the task is processed locally in vehicle v; otherwise, it is 0. $\boldsymbol{x}_{\boldsymbol{mnk}}(\boldsymbol{t}) = \mathbf{1}$ indicates that BS $\boldsymbol{m}$ offloads task $\boldsymbol{k}$ to BS $\boldsymbol{n}$; otherwise, it is 0. $\boldsymbol{x}_{\boldsymbol{mmk}}(\boldsymbol{t}) = \mathbf{1}$ indicates that task $\boldsymbol{k}$ can be processed by the same BS m where it is offloaded because it has sufficient requirements to process the task; otherwise, it is 0. $\boldsymbol{x}_{\boldsymbol{mck}}(\boldsymbol{t}) = \mathbf{1}$ indicates that BS $\boldsymbol{m}$ offloads task $\boldsymbol{k}$ to cloud $\boldsymbol{CS}$; otherwise, it is 0.

The particular task's $\boldsymbol{k}$ decision to offload is represented in Eq. (2). It depends on the vehicle task's requirements, resource availability, and network connectivity.

$$\boldsymbol{x}_{\boldsymbol{k}}(\boldsymbol{t}) = \{\boldsymbol{x}_{\boldsymbol{0k}}(\boldsymbol{t}), \boldsymbol{x}_{\boldsymbol{1k}}(\boldsymbol{t}), \ldots, \boldsymbol{x}_{(\boldsymbol{G}+\boldsymbol{U}+\boldsymbol{CS})\boldsymbol{k}}(\boldsymbol{t})\} \tag{2}$$

where $\boldsymbol{x}_{\boldsymbol{ik}}(\boldsymbol{t}) = \begin{cases} \mathbf{1}, & \boldsymbol{task\ k\ is\ offloaded\ to\ server\ i}, \boldsymbol{i} \in \mathbf{R}\} \\ \mathbf{0}, & \boldsymbol{otherwise} \end{cases}$. $\boldsymbol{x}_{\boldsymbol{0k}}(\boldsymbol{t}) = \mathbf{1}$ indicates that task k is processed locally; otherwise, it is 0.

The task can be executed in one among the following, locally or in any one edge server or the cloud, and it is represented by the following constraint in Eq. (3).

$$\sum_{\boldsymbol{i}=\mathbf{0}}^{\boldsymbol{G}+\boldsymbol{U}+\boldsymbol{CS}} \boldsymbol{x}_{\boldsymbol{ik}}(\boldsymbol{t}) = \mathbf{1} \tag{3}$$

A task is considered to fail if it cannot be processed within its latency constraint $\boldsymbol{l}_{\boldsymbol{k}}$.

Communication Model: The BS in the considered environment can be defined as $\boldsymbol{BS} = \{\boldsymbol{G}, \boldsymbol{U}, \boldsymbol{ST}\}$. The vehicle $\boldsymbol{i}$ and BS $\boldsymbol{j}$ uplink transmission rate $\boldsymbol{r}_{\boldsymbol{ij}}$ is given by Eq. (4).

$$\boldsymbol{r}_{\boldsymbol{ij}} = \boldsymbol{BW}_{\boldsymbol{ij}} \mathbf{log}_{\mathbf{2}}\left(\mathbf{1} + \frac{\boldsymbol{p}_{\boldsymbol{i}}\boldsymbol{g}_{\boldsymbol{ij}}}{\boldsymbol{\sigma}^{\mathbf{2}}}\right) \tag{4}$$

where $\boldsymbol{BW}_{\boldsymbol{ij}}$ is the channel bandwidth, $\boldsymbol{p}_{\boldsymbol{i}}$ is the vehicle $\boldsymbol{i}$'s transmit power, $\boldsymbol{g}_{\boldsymbol{ij}}$ represents the vehicle $\boldsymbol{i}$ channel gain within BS $\boldsymbol{j}$ coverage and $\boldsymbol{\sigma}^{\mathbf{2}}$ is the additive Gaussian noise. The vehicle-to-satellite channel gain can be disregarded since a vehicle and a satellite are separated by a long distance. Table 2 describes the various notations used.

Table 2: Notations and descriptions

Notation	*Description*
c_v	The computing capacity of the Vehicle
c_s	The computing capacity of the Server
d_k	Task data size
c_k	The computation required by the task
$r_{vm}(t)$	The transmission rate of vehicle v to BS m
$r_{vn}(t)$	The transmission rate of vehicle v to BS n
$r_{vc}(t)$	The transmission rate of vehicle v to Cloud **CS**
$delay_{vk}(t)$	Delay of task k processing in vehicle v
$delay_{mmk}(t)$	Delay of task k processing in BS m

(Continued)

Table 2 (continued)

Notation	*Description*
$delay_{mnk}(t)$	Delay of task k processing in BS n
$delay_{mck}(t)$	Delay of task k processing in Cloud **CS**
$\boldsymbol{v_{ct}}$	Vehicle coverage time

Delay Model: The number of tasks offloaded from the vehicles covered by the particular BS $\boldsymbol{m}$ in time slot $\boldsymbol{t}$ is $\boldsymbol{n_m(t)}$. The offloaded tasks at the BS can be processed at various locations, such as locally in the vehicle, the offloaded BS, the nearby BS, or the cloud, depending on the delay constraint of the task and the computational capabilities of the edge servers. The delay calculation for task execution at various locations is discussed below:

(i) Processing locally in the vehicle $\boldsymbol{v}$: The tasks with limited computational requirements can be processed locally in the resource-constrained vehicle. It is assumed that a vehicle generates only one task at a time. The task execution delay comprises only the computing delay and it can be formulated as in Eq. (5).

$$\boldsymbol{delay_{vk}(t) = \frac{d_k * c_k}{c_v}} \tag{5}$$

where $\boldsymbol{c_v}$ represents the computational capacity of a vehicle $\boldsymbol{v}$.

(ii) Processing in the offloaded BS $\boldsymbol{m}$: The number of tasks that are ready for execution in BS $\boldsymbol{m}$ is $\sum_{k=1}^{n_m(t)} \boldsymbol{x_{mmk}(t)}$. In this case, the task execution delay comprises the transmission delay and computing delay and is formulated as in Eq. (6).

$$\boldsymbol{delay_{mmk}(t) = \frac{\sum_{k=1}^{n_m(t)} x_{mmk}(t) * d_k * c_k}{c_s} + \frac{d_k}{r_{vm}(t)}} \tag{6}$$

where $\frac{d_k}{r_{vm}(t)}$ represents the transmission delay of task $\boldsymbol{k}$.

(iii) Processing in the nearby BS $\boldsymbol{n}$: The number of tasks that are ready for execution in BS $\boldsymbol{n}$ is $\sum_{k=1}^{n_n(t)} \boldsymbol{x_{mnk}(t)}$. In this case, the task execution delay comprises the transmission delay and computing delay and is formulated as in Eq. (7).

$$\boldsymbol{delay_{mnk}(t) = \frac{\sum_{k=1}^{n_n(t)} x_{mnk}(t) * d_k * c_k}{c_s} + \frac{d_k}{r_{vn}(t)}} \tag{7}$$

where $\frac{d_k}{r_{vn}(t)}$ represents the transmission delay of task $\boldsymbol{k}$.

(iv) Processing in the cloud: The cloud is assumed to have vast computational resources, with an aggregate processing power ranging from 50 to 500 GHz. Given this immense capacity, it is presumed to be sufficient for executing vehicular tasks without incurring any waiting delays. In this case, the task execution delay comprises the transmission delay and computing delay and is formulated as in Eq. (8).

$$\boldsymbol{delay_{mck}(t) = \frac{d_k * c_k}{c_s} + \frac{d_k}{r_{vc}(t)}} \tag{8}$$

where $\frac{d_k}{r_{vc}(t)}$ represents the transmission delay of task $\boldsymbol{k}$.

The delay of task execution at various destinations is computed as above. However, it's important to acknowledge that not all components within the IoV system may be legitimate; there could also be adversaries. Thus, to protect the privacy of vehicle data and offloading inference attacks, a privacy-preserving technique has to be incorporated. Therefore, the adversary model has also been considered.

Adversary Model: In the vehicular environment, vehicles are susceptible to internal attackers, often called "honest-but-curious" adversaries. As it is an insider, it may have more access to information about vehicles. These attackers behave as legitimate entities but exploit their access to extract sensitive information. The specific threats that may occur are outlined as follows:

- The attacker has access to the trained HetGNN model deployed on its server. This allows the attacker to observe its structure, outputs, and associated metadata.
- The attacker can utilize prior knowledge about the vehicular network (e.g., patterns of communication, typical task offloading preferences) combined with insights from the HetGNN model to deduce information about vehicles in the system.
- By analyzing the HetGNN model, the attacker may infer sensitive vehicle-specific information, such as: Offloading preferences (e.g., which servers a vehicle prefers to use for task computation). Behavioral patterns and locations of vehicles, potentially leading to breaches in location privacy or tracking.

These defined adversarial scenarios underscore the necessity of incorporating robust privacy-preserving mechanisms, such as differential privacy, in the proposed HetGNN framework.

According to the above-explained models, the optimization objective for task offloading is formulated as below.

3.3 Problem Formulation

The proposed novel secure unsupervised HetGNN task offloading algorithm makes the desired offloading decision for the requested task from the vehicles. Thus, the task offloading problem can be considered a decision-making problem. Minimizing the total task execution delay with the optimal task offloading decision is considered the optimization objective. The execution delay of a task $\boldsymbol{k}$ in a particular BS $\boldsymbol{m}$ in a time slot $\boldsymbol{t}$ is obtained by combining the task execution delays of various possible offloading destinations and is expressed as in Eq. (9).

$$\boldsymbol{delay}_{\boldsymbol{mk}}(\boldsymbol{t}) = \boldsymbol{delay}_{\boldsymbol{vk}}(\boldsymbol{t}) + \boldsymbol{delay}_{\boldsymbol{mmk}}(\boldsymbol{t}) + \boldsymbol{delay}_{\boldsymbol{mnk}}(\boldsymbol{t}) + \boldsymbol{delay}_{\boldsymbol{mck}}(\boldsymbol{t}) \tag{9}$$

The overall execution delay of all tasks in a particular time slot t is formulated as in Eq. (10).

$$\boldsymbol{delay}(\boldsymbol{t}) = \sum_{\boldsymbol{m}\in\boldsymbol{BS}} \sum_{\boldsymbol{k}\in\boldsymbol{TD}} \boldsymbol{delay}_{\boldsymbol{mk}}(\mathbf{t}) \tag{10}$$

Formally, the problem of overall task execution delay minimization can be expressed as in Eq. (11).

$$\mathbf{min}\ \boldsymbol{delay}(\boldsymbol{t}) = \sum_{\boldsymbol{m}\in\boldsymbol{BS}} \sum_{\boldsymbol{k}\in\boldsymbol{TD}} \boldsymbol{delay}_{\boldsymbol{mk}}(\mathbf{t}) \tag{11}$$

subject to the following constraints:

$$\sum_{\boldsymbol{i}=\mathbf{0}}^{\boldsymbol{G}+\boldsymbol{U}+\boldsymbol{CS}} \boldsymbol{x}_{\boldsymbol{ik}}(\boldsymbol{t}) = \mathbf{1}\forall \boldsymbol{k} \in \boldsymbol{TD}, \forall \boldsymbol{i} \in \boldsymbol{R} \tag{12}$$

$$\boldsymbol{delay}_{\boldsymbol{mk}}(\mathbf{t}) \leq \boldsymbol{l}_{\boldsymbol{k}}\boldsymbol{m} \in \boldsymbol{BS}, \boldsymbol{k} \in \boldsymbol{TD} \tag{13}$$

$$\boldsymbol{delay}_{\boldsymbol{mk}}(\mathrm{t}) \leq \boldsymbol{v}_{\boldsymbol{ct}}\boldsymbol{m} \in \boldsymbol{BS}, \boldsymbol{k} \in \boldsymbol{TD} \tag{14}$$

where l_k represents the latency constraint of task k and v_{ct} represents the vehicle coverage time, i.e., the duration of the vehicle with its associated BS before moving out of the communication range. The objective in Eq. (11) aims to minimize the total execution delay of all the tasks in the SAGIVN environment. The constraint in Eq. (12) is used to ensure that task k of BS m can opt for only one option from local processing, processing in the corresponding BS m, offloading to another BS n for processing, or offloading to the cloud for processing. The constraints in Eqs. (13) and (14) are introduced to ensure that the delay of processing tasks should not exceed the delay tolerance time and should be less than the vehicle coverage time to return the vehicle task results to the vehicle before it moves out of the corresponding BS coverage area.

The conventional approach, which generates offloading decisions through iteration, suffers from high computational complexity, and as the number of edge servers and vehicles increases, the solution time escalates correspondingly. In the dynamic environment of the IoV, swift acquisition of the offloading strategy becomes paramount. To enhance the QoS and to provide privacy preservation for the task offloading challenge in a dynamic environment, we propose a secure unsupervised HetGNN algorithm. The optimization objective in Eq. (9) is incorporated as the loss function for the unsupervised HetGNN, thereby making the optimal task offloading decision with reduced delay.

4 GNN-Based Framework for Task Offloading in the IoV

IoV networks can be naturally represented as graphs, making Graph Neural Network a suitable choice for modeling and processing them. GNN increases its applicability for IoV networks by capturing spatial and temporal context as well as supporting scalability and adaptive decision-making. Thus, to achieve optimal computational offloading decisions that minimize the delay as defined in Eq. (9), we design and implement a GNN-based approach as follows.

4.1 Graph Representation of the IoV

The IoV environment is represented as a heterogeneous graph where vehicles, edge servers associated with UAVs, & RSUs/BSs, cloud server, and satellite act as the nodes in the graph. An edge is drawn from node i to node j if there exists a communication link between them. Since the nodes are of various types, a heterogeneous graph is considered. Vehicular nodes are characterized by features such as task attributes (data size, computational requirements, and latency constraints), while server nodes, including edge servers in RSUs, base stations, UAVs, and cloud servers, are characterized by computational resources as features. The edge weights in the graph are determined by the transmission rate between the nodes, reflecting the data transfer capacity of the communication links. The graph generated by the IoV network topology is called an IoV graph. Formally, an IoV graph is an ordered tuple $G = (N, E)$, where nodes (vehicles and servers) are represented by N, edges by E. The feature matrix F consists of the node attributes, where $F = \{F_v, F_s\}$, with $F_v \in \mathbb{R}^{VX3}$ denoting the vehicle node feature matrix and $F_s \in \mathbb{R}^{RX1}$ representing the server node feature matrix. The vehicle node features comprise of task's data size d, task's computation requirement c, and task's latency constraint l expressed as $F_v = [d, c, l]$. The server node feature includes computational resource c_s, given by $F_s = [c_s]$. Here, V is the number of vehicles, and R is the number of servers. The adjacency array is represented as $AA \in \mathbb{R}^{(V+R)X(V+R)}$ and is given in Eq. (15).

$$AA_{(i,j)} = \begin{cases} f_{ij}, & (i \in \mathbf{V} \wedge j \in \mathbf{R}) \vee (i \in \mathbf{R} \wedge j \in \mathbf{V}), \\ \mathbf{0}, & otherwise \end{cases} \tag{15}$$

where $\mathbf{V}$ and $\mathbf{R}$ denote the sets of vehicles and servers in the graph, respectively, and f_{ij} represents the existence of edges associated with the nodes. The different types of nodes in this graph are vehicles, UAVs

associated with the MEC server, RSUs/BSs associated with the MEC server, the cloud server, and satellite. The vehicles that are not in the communication range of any ground or air base station is connected to the satellite. Here, satellite act as the relay node. The edge types include vehicle to server, server to server, server to vehicle, server to cloud, and cloud to server. This graph structure is suitable for capturing the interactions between vehicles and servers as well as between servers and servers. The representation of the IoV environment as a heterogeneous graph is done, making them suitable for modeling diverse entities and relationships in the IoV environment. Fig. 2 shows a graph representation of the IoV environment.

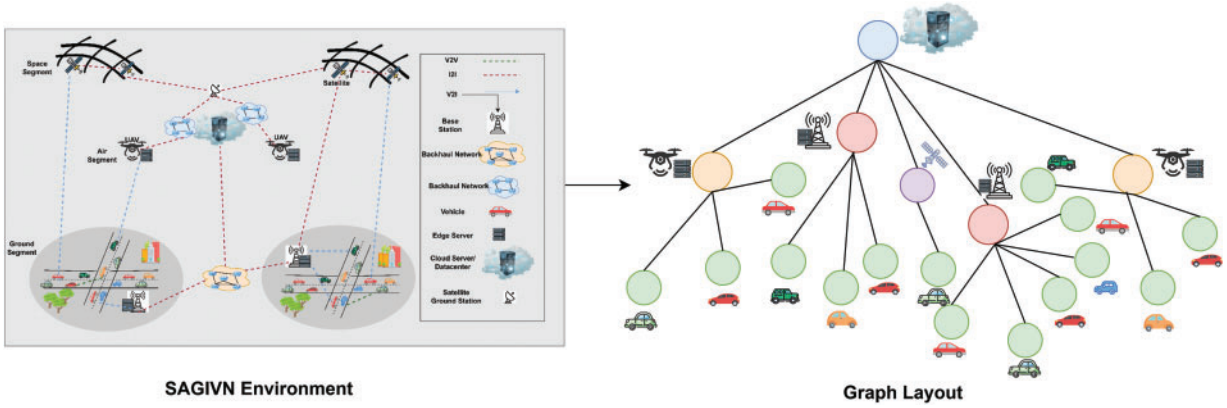

Figure 2: Graph representation of the IoV environment

4.2 Design of the HetGNN Algorithm

GNNs have expanded upon existing neural network architectures, enabling them to analyze structured data in the form of graphs [24]. In the GNN, a multilayer structure exists to enable each node to aggregate its features from its neighbors and combine them with its features at each layer. The GNN helps in exploring the spatial features of the IoV environment. A novel scalable heterogeneous GNN architecture, HetGNN, is developed for task offloading problems. Fig. 3 represents an overview of the HetGNN-based computation offloading.

An attention-based message passing graph neural network captures the neighbor features among nearby neighbor nodes. The aggregation of features occurs at each stage of message passing, which is accumulated as a particular node's embedding of graph G. The proposed heterogeneous GNN consists of an encoder layer, GNN message passing layers, and a task allocation multilayer perceptron (MLP) layer. Fig. 4 depicts the proposed HetGNN architecture.

The vehicle and server node embeddings are extracted using the encoder with feedforward neural network layers. Message passing, allows nodes to exchange information with their nearby nodes to perform aggregation and updation of features. Message-passing layers are used to process the task information from vehicles to the server and to process the offloading information from the server to the vehicle. Here, linear transformations are applied to the features of nodes and edges for the extraction of relevant information that may help in capturing meaningful representations of features. By allowing the nodes to propagate the information through the graph structure, the GNN model can learn the representations by considering the relationships between the nodes, thus enabling them to make decisions regarding task offloading.

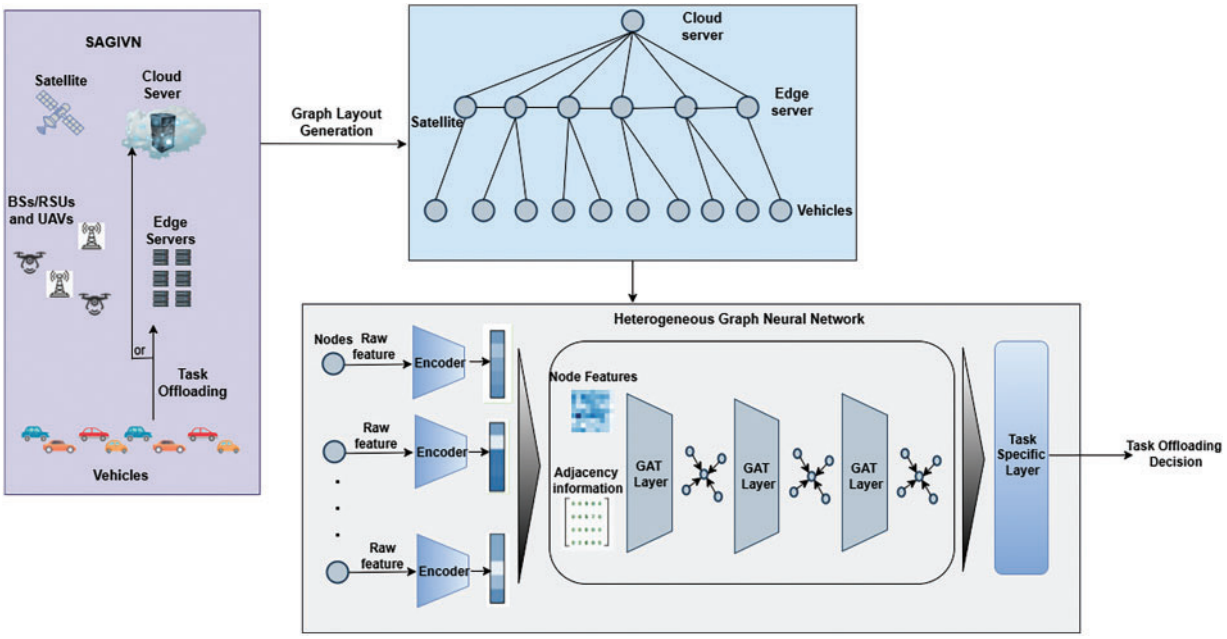

Figure 3: An overview of the HetGNN-based computation offloading

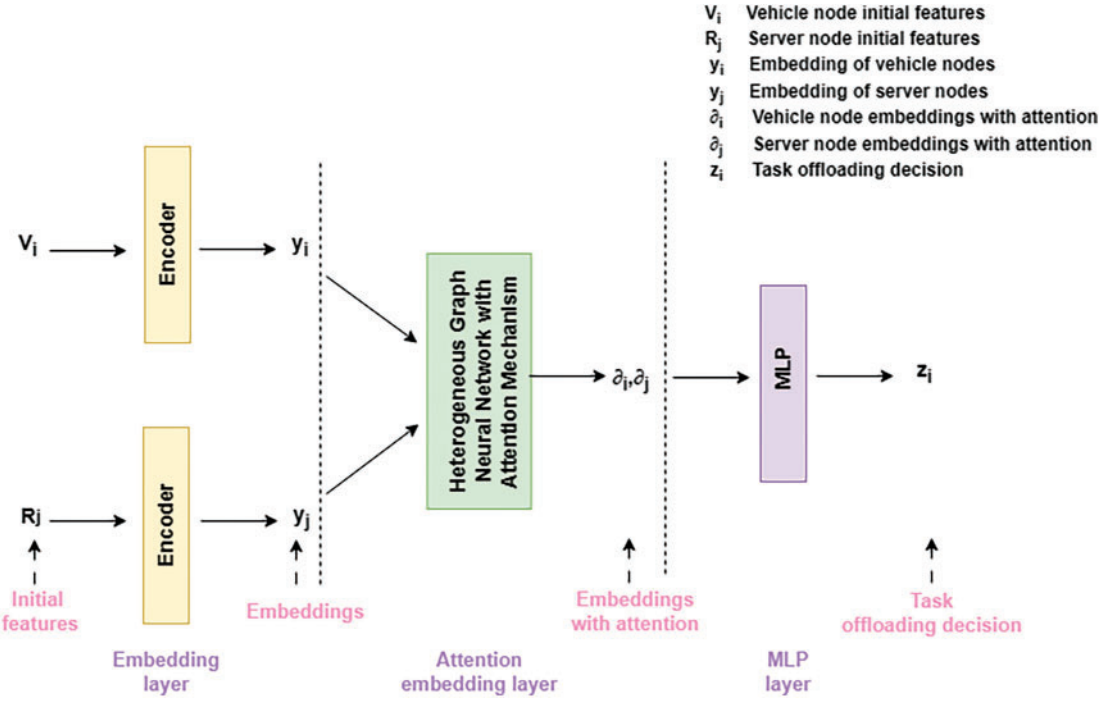

Figure 4: HetGNN architecture

All node or edge features may not be of the same importance to all other nodes. An attention mechanism is introduced to assign varying degrees of importance to various nodes and edges. The contributions of different nodes and edges in the graph are weighed using the attention mechanism in the proposed work. A task with very low latency cannot be offloaded to the server in other BSs or the cloud. In such cases, attention

must be given to local computing or offloading to the server of the current BS to avoid task failure. With the message attributes from the forward pass of the message passing layers, the optimal task offloading decision is made using the MLP layer. The aggregation and attention mechanisms are explained as follows.

Aggregation: The aggregation phase of the GNN involves the mean and updation steps over the neighbor-provided messages. The mean $\boldsymbol{M}$ of the particular node's $\boldsymbol{i}$ neighbor nodes $\boldsymbol{N(i)}$ is calculated using a $\boldsymbol{message_{ij}} = \sum_{j \in N(i)} \boldsymbol{M}(\boldsymbol{y_i}, \boldsymbol{y_j})$, where $\boldsymbol{y_i}$ represents the node's embeddings and $\boldsymbol{y_j}$ represents the neighbor node's embeddings. Then, the embeddings of the particular node, $\boldsymbol{message_{ij}}$, are updated based on the sum of the neighbor node's embeddings and its embeddings. $\boldsymbol{U}$ represents the sum function that outputs the updated embeddings of the node according to equation $\boldsymbol{y_i} = \boldsymbol{U}(\boldsymbol{message_{ij}}, \boldsymbol{y_i})$. In a graph, all nodes' embedding values are represented as a vector that is collected during the aggregation phase of the GNN.

Attention: The attention mechanism in GNNs enables them to learn a dynamic and adaptive local summary of neighborhood nodes. It alternates between a propagation layer and an attention layer. The hidden states of neighborhood nodes are aggregated, and attention weights are computed for each neighborhood node. Then, the current node features are updated based on the weighted sum of neighbor node features. This ensures that the model learns to focus on the important information while downplaying the less relevant neighbors.

Attention weights are calculated to assign varying importance to different nodes when generating messages. The attention weights are obtained using the attention scores. A set of learnable parameters based on the features of the nodes is used to calculate the attention scores. $\boldsymbol{a_{ij}}$ represents the attention score for a pair of nodes. The updating function of message aggregation $\boldsymbol{y_i} = \boldsymbol{U}(\boldsymbol{message_{ij}}, \boldsymbol{y_i})$ is rewritten as in Eq. (16) with the incorporation of the attention mechanism.

$$\boldsymbol{message_{ij}} = \boldsymbol{AGG}\left(\boldsymbol{y_i}, \boldsymbol{y_j}\right), \boldsymbol{AA_{(i,j)}} \neq \boldsymbol{0} \tag{16}$$

$$\partial_{\boldsymbol{i}} = \delta(\boldsymbol{y_i}, \varphi\{\boldsymbol{\alpha_{ij}} \boldsymbol{message_{ij}} : \boldsymbol{j} \in \boldsymbol{Neighbor(i)}\}), \boldsymbol{i}, \boldsymbol{j} \in \boldsymbol{V} \vee \boldsymbol{R} \tag{17}$$

$$\boldsymbol{\alpha_{ij}} = \frac{\boldsymbol{exp}(\boldsymbol{a_{ij}})}{\sum_{f \in \boldsymbol{Neighbor(i)}} \boldsymbol{exp}(\boldsymbol{a_{if}})} \tag{18}$$

where $\boldsymbol{V}$ and $\boldsymbol{R}$ represent the node sets of vehicles and servers, respectively. In Eq. (16), $\boldsymbol{message_{ij}}$ represents the extracted message passed from node $\boldsymbol{i}$ to $\boldsymbol{j}$. $\boldsymbol{AGG}(.)$ represents the trainable aggregation network that extracts the features of neighboring nodes. In Eq. (17), $\varphi(.)$ represents the message concatenation function which compresses the message from all adjacency nodes to a vector. $\boldsymbol{\alpha_{ij}}$ represents the attention value that identifies the amount of influence of different neighboring nodes on the particular node $\boldsymbol{i}$ and is obtained as in Eq. (18). $\boldsymbol{Neighbor}\,(\boldsymbol{i})$ is the set of node $\boldsymbol{i's}$ neighbor nodes. δ represents the trainable neural network that is used to update node features based on the initial input features $\boldsymbol{y_i}$, to the output features $\boldsymbol{\partial_i}$.

Attention can be crucial for dynamically adapting to changing conditions, varying computational capabilities, and communication patterns among vehicles and servers in the context of IoV task offloading. The attention mechanism can contribute to effective task-offloading decision-making by enhancing the ability of the GNN to capture and utilize the superlative node for task-offloading with the relevant information from the graph. Utilizing message passing, attention mechanisms, and an MLP, the proposed HetGNN algorithm implements the task offloading strategy and undergoes unsupervised training. The HetGNN is trained on diverse graph structures with varying node characteristics. This enables HetGNN to generalize across different topologies and process graphs with varying vehicular nodes and communication links. Consequently, HetGNN can adapt to topology changes with a varying number of vehicles, ensuring optimal task-offloading decisions.

4.3 Unsupervised HetGNN Training

The problem of task offloading in the IoV environment cannot be solved using supervised learning methods, as there is no defined set of datasets available. To make the proposed HetGNN learn about the efficient task offloading mechanism in an unsupervised manner, an appropriate loss function must be defined to propagate the error in the training process. The loss function is defined concerning the objective function of the task offloading problem as defined in Eq. (9). The objective is to minimize the overall task execution delay by considering the communication and computation delays. The loss function can be formulated as the sum of the communication time and computation time. The aggregated loss is calculated as the meantime loss of all vehicles. Table 3 shows the training process of the proposed algorithm.

Table 3: Secure unsupervised HetGNN task offloading algorithm

Secure unsupervised HetGNN task offloading algorithm

Input: hetero-data, initial model weights, learning rate, privacy level
Output: learned HetGNN model with updated weights
for epochs 1 to e do
for batches 1 to b_s do
$\boldsymbol{y}_i, \boldsymbol{y}_j$ = Encoder($\boldsymbol{V}_i, \boldsymbol{R}_j$)
$\boldsymbol{\partial}_i$ = Message Passing Layer ($\boldsymbol{y}_i, \boldsymbol{y}_j$); $\boldsymbol{j} \in \boldsymbol{Neighbor(i)}$
$\boldsymbol{y}_{ij}$ = MLP Layer ($\boldsymbol{\partial}_i$)
loss = delay *#loss calculation*
loss.backward() *#backward propagation of loss*
noise = ε/b_s * no. of parameters *#noise calculation*
param.grad += noise *#addition of noise to gradients*
optimizer.step() *#update* via *stochastic gradient descent*
end
end

$$\textbf{Loss} = \textbf{Communication time} + \textbf{Computation time}; \textbf{Loss} = \boldsymbol{delay}_{mk}(\boldsymbol{t}) \tag{19}$$

$$\textbf{Computation time} = \frac{\textbf{Data size}}{\textbf{Computation resource allocated}}; \textbf{Computation time} = \frac{\boldsymbol{d_k * c_k}}{\boldsymbol{c_s}} \tag{20}$$

$$\textbf{Communication time} = \frac{\textbf{Data size}}{\textbf{Transmission rate}}; \textbf{Communication time} = \frac{\boldsymbol{d_k}}{\boldsymbol{r_{vm}(t)}} \tag{21}$$

By training with the above loss function in Eq. (19), the proposed algorithm learns to minimize the overall execution time of tasks, thereby reducing the task failure rate in the dynamic IoV environment. Eqs. (20) and (21) represent the computation and communication times, respectively. The vehicular task requirements and server's computing capacity are considered as node features in HetGNN. Through aggregation and attention mechanism, the features are analyzed and the appropriate offloading decision is made using delay as the loss function. Table 4 presents the execution flow of the HetGNN algorithm. Adversaries attempt to glean information from the algorithm, hence DPSGD is employed during the training of the proposed algorithm.

Table 4: Execution flow of the HetGNN algorithm

Execution flow of HetGNN algorithm
do
The Controller receives information from the BSs and vehicles within its control
Generate a graph layout with the gathered information of vehicles, RSUs edge servers, UAVs edge servers, cloud server
The proposed mechanism in the BS makes an optimal task-offloading decision
Send the task to the edge server for execution
while not end of the timeframe

4.4 Privacy Preserving Unsupervised HetGNN Training

With prior knowledge about the IoV environment and from the trained HetGNN model and by observing its behavior, adversaries may try to infer information about the vehicle's location as well as offloading preference. Thus, differential privacy-based techniques shall be incorporated to preserve privacy. This can be done by incorporating DPSGD into HetGNN's training.

Differential Privacy Stochastic Gradient Descent Training (DPSGD) is designed by combining the principles of differential privacy with stochastic gradient descent [39]. It involves adding a controlled level of noise to the gradients computed during each iteration of the training process. It ensures the model does not reveal sensitive information about individual data points, even if an adversary can access the trained model and its parameters. The noise is typically drawn from a Gaussian probability distribution and is added to the gradients of the parameter. Thus, the parameter update $\boldsymbol{\theta_{i+1}}$ in the DPSGD mechanism occurs as in Eq. (22).

$$\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i} - \boldsymbol{\xi}(\boldsymbol{g_i}(\boldsymbol{\theta_i}; \boldsymbol{D}) + \boldsymbol{N}\left(\boldsymbol{0}, \boldsymbol{\sigma^2}(\boldsymbol{\varepsilon})\boldsymbol{I_d}\right)) \tag{22}$$

As in Eq. (22), $\boldsymbol{N}\left(\boldsymbol{0}, \boldsymbol{\sigma^2}(\boldsymbol{\varepsilon})\boldsymbol{I_d}\right)$ Gaussian noise distribution is added to the $\boldsymbol{g_i}$ gradients of the parameter $\boldsymbol{\theta_i}$. $\boldsymbol{D}$ represent the data, $\boldsymbol{\sigma^2}$ represents the noise variance, $\boldsymbol{\xi}$ is the learning rate, and $\boldsymbol{\varepsilon}$ is the privacy budget. The level of noise magnitude is controlled by a privacy parameter denoted as epsilon ε, referred to as privacy budget. Without noise, an attacker with access to model updates could reverse-engineer the training data by analyzing the gradient updates. A smaller value of ε can guarantee stronger privacy; however, this may have a considerable impact on the utility of our learned HetGNN. Thus, the noise must be carefully calibrated to ensure that individual training data points do not excessively influence the model's updates. An informed decision about the level of privacy protection is required to control the trade-off between privacy and system utility.

Based on the observed updates to the model, it will be more challenging for attackers to extract sensitive information about individual data points because of the introduction of noise by DPSGD. An adversary may try to infer a vehicle's offloading decision based on its observed behavior. With this information, the attacker can trick the vehicles into considering malicious BSs for offloading. This issue shall also be mitigated by integrating DPSGD. As this guarantees that the adversary cannot reconstruct the individual vehicle offloading patterns, making the offloading destination prediction indeterministic.

Theorem 1: *Privacy Preservation of Vehicular Data Using DPSGD in HetGNN*

Let $\boldsymbol{A}$: $\boldsymbol{\chi} \rightarrow \boldsymbol{R}$, where χ is the domain and $\boldsymbol{R}$ is the range, be a differentially private learning algorithm that trains the model using Differentially Private Stochastic Gradient Descent. For any two neighboring datasets

$D_1, D_2 \in \chi$ differing by at most one data record, the output of the learning algorithm satisfies (ε,δ)-differential privacy, ensuring that no adversary can infer specific vehicular data with high confidence

Proof:

To prove that the training mechanism preserves privacy, we rely on the principles of differential privacy, which ensures that the inclusion or exclusion of any single data does not significantly change the model's output distribution.

A learning algorithm A satisfies (ε, δ)—differential privacy if, for all measurable sets M and for all neighboring datasets D_1 and D_2, we have: $p[A(D_1) \in M] \leq exp(\varepsilon)\, p[A(D_2) \in M] + delta$. This definition ensures that the probability of obtaining any specific output from dataset D_1 is almost the same as from D_2, making it difficult for an adversary to distinguish whether a particular vehicular environment was used in training. In DPSGD, privacy is enforced by modifying the standard stochastic gradient descent update rule. In DPSGD, controlled noise is added to each gradient update to obscure the contribution of any individual data point: $\theta_{i+1} = \theta_i - \xi(g_i(\theta_i; D) + N(0, \sigma^2(\varepsilon) I_d))$, where $N(0, \sigma^2(\varepsilon) I_d)$ is Gaussian noise with variance σ^2. The addition of noise ensures that even if an attacker observes the model updates, they cannot accurately infer whether a particular vehicle's data was used in training. The noise is adjusted based on the sensitivity of the gradients and the specified privacy budget ε value.

To mitigate the impact of training samples on the parameters, the gradients are bounded by the L_2 norm: $\Pi(g_i) = g_i.min(1, C/||g_i||_2)$, where g_i corresponds to the gradient of the ith sample and C is the clipping factor. By restricting gradient magnitudes, we ensure that no individual data point can have an excessive impact on the model updates, thereby reinforcing privacy.

It has been demonstrated in [40] that each step of DPSGD achieves (ε, δ)—differential privacy when the noise parameter $\sigma = Cz$ is adjusted as follows: $z = \frac{\sqrt{2\ln\frac{1.25}{\delta}}}{\varepsilon}$.

This ensures that the probability distribution of gradients remains indistinguishable regardless of whether any single data is present in the dataset. By setting appropriate values of ε and δ, we ensure that even across multiple training updates, an adversary cannot infer with high confidence whether a particular data is contributed to the model. Thus, by introducing Gaussian noise and enforcing gradient clipping, individual data remains indistinguishable within the dataset, ensuring that an adversary cannot extract sensitive information about specific vehicles and its offloading pattern. Hence, HetGNN with DPSGD preserves vehicular data privacy, proving the theorem. □

5 Experimental Results and Performance Analysis

The proposed privacy-preserving unsupervised HetGNN task offloading framework is evaluated in a simulation environment developed using Python 3.8.18, leveraging libraries such as PyTorch 2.1.0 and Deep Graph Library (DGL) to implement and train the GNN models. The experiments were conducted on a hardware setup comprising a 64-bit Windows OS, 64 GB RAM, an NVIDIA Quadro P5000 GPU with 16 GB memory, and an Intel Xeon processor. The SAGIVN architecture is represented as a heterogeneous graph, and its parameter values are selected randomly within the ranges outlined in the Table 5. The HetGNN is trained using a synthetically generated dataset comprising 2048 graph layouts, with characteristics varying in the range as in Table 5. The dataset consists of graphs with varying numbers of nodes, ensuring the model's scalability to support vehicle numbers varying during regular and peak traffic hours as well as varying amount of edge servers. The synthetically created dataset was split into an 80:20 ratio for training and testing the HetGNN model. The performance of the proposed HetGNN is evaluated using parameters such as task execution delay, task failure rate, inference time and resource utilization. HetGNN's results were compared with various baseline methods, including Genetic Algorithm (GA), Deep Neural Network (DNN), Deep Q

Learning (DQN), and Deep Deterministic Policy Gradient (DDPG). The findings demonstrate that HetGNN achieved a 13%–50% reduction in system delay, a 30%–50% reduction in task failure rate, and a significantly lower inference time of 0.21 s compared to the baseline methods. These results highlight the statistically significant improvements in the key performance metrics achieved by HetGNN, demonstrating its efficacy and suitability for task offloading in SAGIVN environments.

Table 5: Parameter settings

Parameter	Value
No. of vehicles	25, 35, 45, 55, 65
No. of servers	14, 16, 18, 20, 22
Vehicles Transmission range	250–300 m
BSs Transmission range	1 km
Vehicle speed	40–130 km/h
Bandwidth ($\boldsymbol{BW}$)	40 MHz
Transmission power ($\boldsymbol{p_i}$)	30 dBM
Gaussian noise ($\boldsymbol{\sigma^2}$)	−100 Dbm
The computing capacity of servers ($\boldsymbol{c_s}$)	[5, 10, 15, 20, 25], [50–500] GHZ
The computing capacity of vehicles ($\boldsymbol{c_v}$)	2 GHz
Size of computation tasks ($\boldsymbol{d}$)	[10, 20, 30, 40, 50] Mb
CPU cycles required by tasks ($\boldsymbol{c}$)	N(1000, 10) Megacycles
Delay Constraint ($\boldsymbol{l}$)	0.5–2.0 s
No. of epochs (e)	300
Privacy level (ε)	0.2

5.1 Generation of Training Samples

One of the key challenges in the IoV is designing the network layout considering the spatial arrangement of vehicles and servers in the IoV environment. The process of creating realistic representations of the IoV environment can be done by generating diverse layouts that capture the dynamic distribution of vehicles and servers within a specified area. Layouts for different IoV environments are generated based on the given number of vehicles and servers. To illustrate SAGIVN, vehicles, edge servers in UAVs, RSUs, BSs, and cloud servers are represented as nodes in the graph.

A predetermined number of vehicles with a randomized location within their communication range are generated by iterating over the number of servers. The task size, delay constraint, and computation required are considered vehicle node features, and the computation capability is considered a server node feature. The edges among the nodes are generated based on the communication range of the base stations. The transmission rate of the particular communication link is also calculated based on the values of the carrier frequency, signal coefficient, transmission power, and distances, and it acts as the edge weight. It is important because it affects the quality of the communication between vehicles and servers. To effectively model the interconnected nature of the IoV environment, a heterogeneous graph data structure is used. This supports the representation of different node types and their relationships. The number of vehicles, servers, and their attributes are given as input to construct the graph using the torch library. The generated graphs are inputted into the proposed HetGNN algorithm for processing.

5.2 Simulation Settings

In the IoV environment, we consider the following components: vehicles, RSUs, and UAVs with an edge server and a cloud server. The vehicles are randomly distributed under the coverage range of each BS with a random speed in the range of 40–130 km/h. The vehicles send task data to the BS with a data rate of 4 Mbps. The coverage area of each BS with an edge server is considered to be a 1 km radius with a fixed location for each layout. The computational capacity of each edge server is fixed and limited. In the task model, the following assumptions are made. The CPU cycles required by the task follow a normal distribution and are generated randomly, that is, N(1000,10) megacycles. The delay constraint of each task is set between 0.5 and 2.0 s. Table 5 presents the experimental parameter settings. In addition, in the HetGNN model, the optimizer is Adam, the maximum number of epochs e is 300, the learning rate ξ is 0.001, and the batch size b_s is set to 32. The training and testing data are generated in the form of a graph layout with the above parameters. The training epochs are set as 300 with 2048 types of graph layouts, and the results are analyzed.

5.3 Results Analysis

The results of the proposed secure unsupervised HetGNN task offloading algorithm are evaluated via various experiments and compared with the following benchmark algorithms.

Genetic Algorithm (GA): The GA belongs to the class of evolutionary algorithms and is a metaheuristic mechanism for solving optimization problems with bio-inspired operators such as crossover, selection, and mutation. The fitness value is calculated for the population in each generation and the objective function is mapped with the fitness function. The destination to which the task shall be offloaded is encoded and the corresponding fitness function is generated according to the optimization objective in the equation. The parents are selected to generate the offspring; thus, a new population is created, and the optimal offloading strategy is selected from this population.

Deep Neural Network (DNN): The multilayer perceptron is used to extract the features of edge servers and vehicles to make the offloading decision. It is pretrained with a fixed number of edge servers and vehicles, i.e., a fixed layout, and it cannot be modified because the input and output dimensions for the DNN are fixed numbers. When the number of vehicles varies from the pretrained layout, DNN will only provide optimal performance for a subset of vehicles and will randomly decide on other nodes. The structure of the DNN must be changed, and it must be retrained for every varying layout of the IoV environment.

Deep Reinforcement Learning (DRL): DRL is an unsupervised algorithm that combines reinforcement learning and deep learning. It deals with learning by interacting with the environment to maximize the reward signal. The algorithm optimizes the objective given in the form of a reward with the experience gathered by the trial-and-error method. The computation offloading problem is constructed as a Markov decision process, and then the DQN, value-based DRL algorithm, and DDPG, policy-based DRL algorithm are utilized to achieve the optimal offloading strategy. The results of the analysis of the various parameters of the proposed algorithm are detailed below.

5.3.1 Analysis of the Hyperparameter Effect on the Convergence

The learning rate is the crucial hyperparameter for all deep learning algorithms during the training of neural networks because it determines the number of steps considered for the optimization process. Therefore, the impact of the learning rate in the HetGNN is significant because it may affect the convergence speed, training stability, and model performance. The experiment aims to assess the HetGNN's convergence speed by varying the learning rate. The experimental setup mirrors the simulation settings outlined in Table 5, with a privacy level of $\mu = 0.2$ and learning rates ξ of 0.01, 0.001, and 0.0001. The HetGNN performance shown

in Fig. 5 shows that for $\boldsymbol{\xi}$ = **0.01**, it cannot converge to a particular point appropriately; for $\boldsymbol{\xi}$ = **0.0001**, it converges to a point similar to that of the beginning; for $\boldsymbol{\xi}$ = **0.001**, it converges appropriately. Consequently, an excessively low learning rate can impede convergence, while an excessively high learning rate may hinder the model from converging to the optimal solution, thereby preventing it from reaching the desired value. Thus, the congruous learning rate is the critical aspect of any neural network.

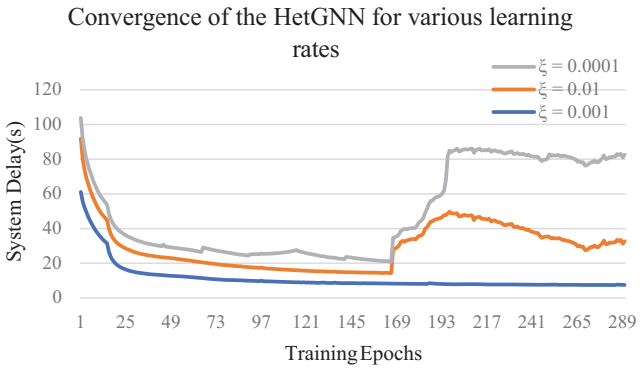

Figure 5: Convergence of the HetGNN for various learning rates

5.3.2 Analysis of the Effect of Privacy Level ε on Convergence

The privacy of the system is ensured by adding a controlled level of Gaussian noise to the model parameters, and the amount of noise to be added depends on the tradeoff between privacy and model performance. An experiment is conducted to evaluate the HetGNN based on the influence of various privacy levels on convergence. The initial model parameters are configured by the simulation setting and Table 5, with a learning rate of $\boldsymbol{\xi}$ = **0**.01. The privacy level ε is then varied as 0, 0.2, 0.32, 0.72, and 1.0024. Setting ε = 0 enables an analysis of the model convergence without differential privacy noise. Fig. 6 reveals that as the privacy level increases, the model's convergence performance gradually subsides. Notably, at ε = 1.0024, the HetGNN exhibits poor performance due to the higher privacy level, where an increased amount of differential privacy noise is introduced during model training. This experimental analysis underscores the necessity of striking a balance between convergence and privacy by setting an appropriate privacy level. Among the privacy levels examined in this experiment, ε = 0.2 provides better performance and is more appropriate.

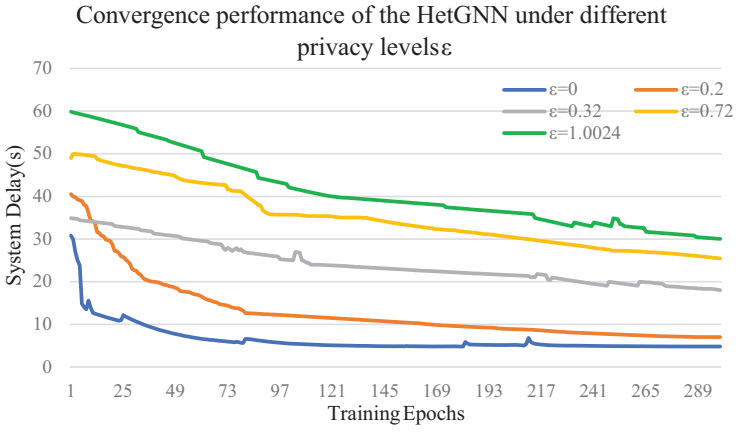

Figure 6: Convergence performance of the HetGNN under different privacy levels ε

5.3.3 Analysis of Different Baseline Methods on the Convergence

To assess the efficacy of the proposed secure unsupervised HetGNN model, baseline methods, including DNN, DQN, and DDPG, are employed, and their performances are compared. The experiment is conducted with the initial parameters the same as in the simulation settings. Fig. 7 illustrates a comparison between the proposed algorithm and the baseline methods regarding their convergence performance. The DNN has issues such as low scalability, and high training costs with the IoV system, as it is difficult to adapt to the varying number of vehicles and edge servers. Hence, it has an unstable lower performance, as shown in Fig. 7. The DQN and DDPG algorithms adapt well to the dynamic IoV system, with large amounts of efficient samples and a balanced exploration-exploitation strategy. However, obtaining better performance is highly time-consuming, challenging, and computationally intensive. As shown in Fig. 7, it fails to converge faster. The experimental results reveal that the proposed secure unsupervised HetGNN algorithm outperforms the DNN, DQN, and DDPG algorithms in terms of convergence speed, suggesting that the HetGNN can adapt to environmental changes with minimal finetuning in resource-constrained edge servers.

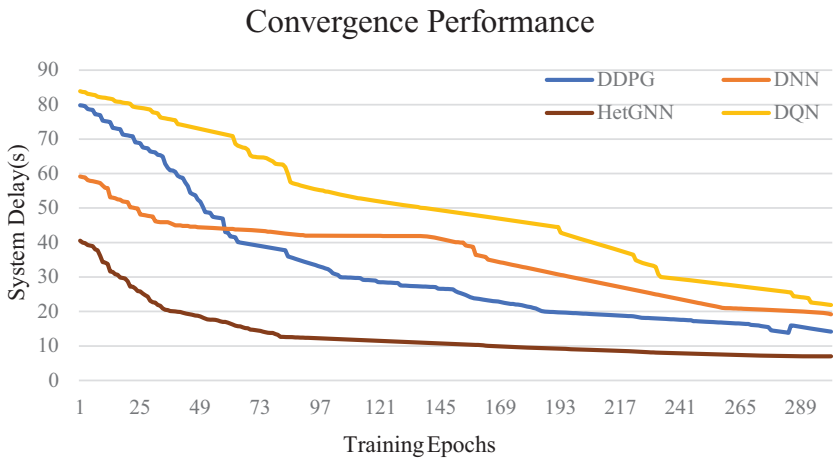

Figure 7: Convergence performance

5.3.4 Analysis of the System Delay across Various Numbers of Vehicles

The proposed algorithm is evaluated with multiple graph layouts with varying numbers of vehicles and 16 edge servers. Fig. 8 depicts the performance assessment of the system delay (i.e., total task execution delay) across various numbers of vehicles in the IoV system. Fig. 8 shows that as the number of vehicles increases, the total execution delay also increases for all algorithms. This is the result of sharing the resources of edge servers with a growing number of vehicles. When there are 25 vehicles, the performances of the secure unsupervised HetGNN are 2.34%, 9.78%, 10.56%, and 24.52% superior to those of DDPG, DQN, DNN, and GA, respectively. Moreover, as the number of vehicles rises from 25 to 65, the HetGNN continues to outperform the baseline methods by 13.13%, 36.41%, 40.96%, and 55.76% compared to DDPG, DQN, DNN, and GA, respectively. This shows that the proposed scheme has good robustness in scenarios with varying numbers of vehicles, thus ensuring scalability and adaptive task offloading decisions in low-vehicle and high-vehicle density scenarios by minimizing the overall task execution delay.

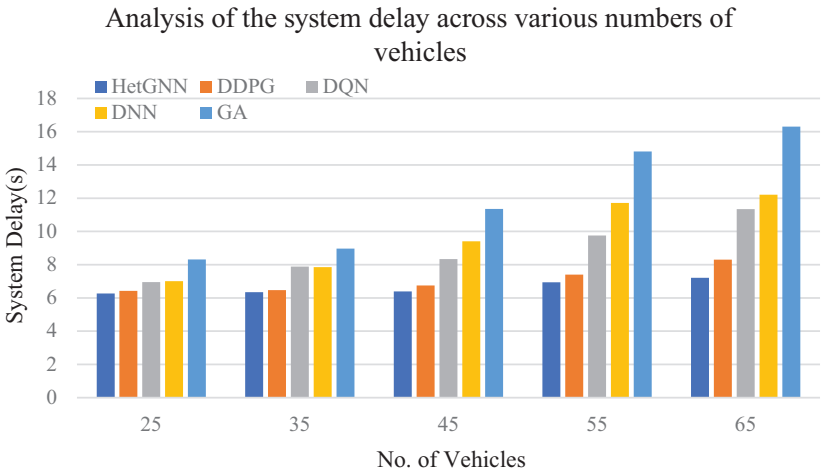

Figure 8: Analysis of the system delay across various numbers of vehicles

5.3.5 Analysis of the System Delay across Various Numbers of Edge Servers

The proposed algorithm is evaluated with multiple graph layouts with varying numbers of edge servers and 55 vehicles. Fig. 9 depicts the performance assessment of system delay (i.e., total task execution delay) across varying numbers of edge servers in the IoV system. Fig. 9 shows that as the number of edge servers increases, the total execution delay decreases for all algorithms. This is because of the increased availability of computational resources as the number of edge servers grows, which can be shared by the vehicle's task for computation. Peculiarly, for 14 edge servers, the HetGNN approach exhibits superior performance compared to DDPG, DQN, DNN, and GA approaches, with improvements of 2.83%, 29.82%, 33.38%, and 42.56%, respectively. Furthermore, an increase in the number of edge servers helps satisfy vehicular task demands, thereby reducing the overall delay. Notably, when the number of edge servers increases to 22, HetGNN outperforms DDPG, DQN, DNN, and GA by 14.39%, 27.67%, 32.55%, and 46.52%, respectively.

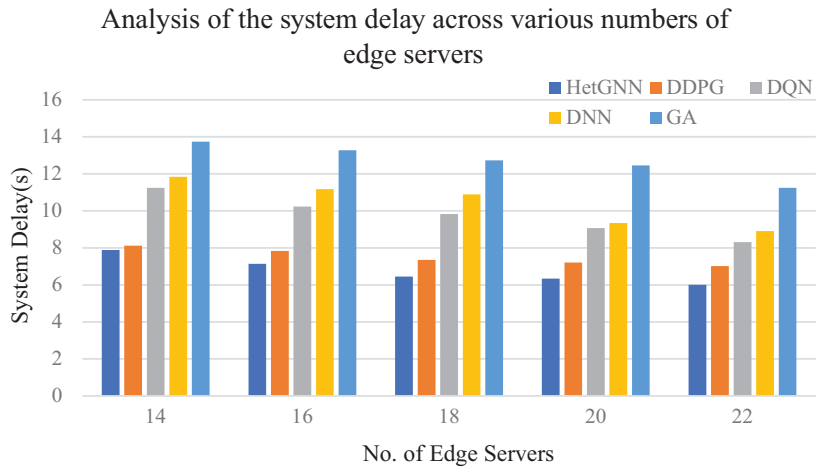

Figure 9: Analysis of the system delay across various numbers of edge servers

5.3.6 Analysis of the System Delay across Various Data Sizes

Fig. 10 presents the analysis of system delay for various algorithms across various data sizes of vehicle tasks. Fig. 10 clearly shows that the system delay escalates with increasing computational task data size. This is attributed to the significant influence of data size growth on transmission time, as well as the corresponding increase in task processing time. HetGNN demonstrates superior performance compared to DDPG, DQN, DNN, and GA as the task magnitude increases. This stems from the HetGNN's ability to make optimal offloading decisions by considering task features and edge server features. Notably, under a data size of 30 MB, the HetGNN outperforms DDPG, DQN, DNN, and GA by approximately 16.96%, 27.72%, 31.62%, and 51.10%, respectively.

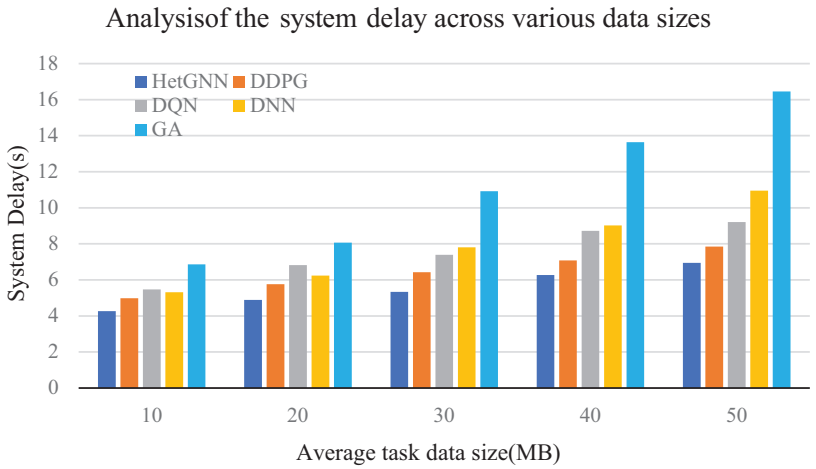

Figure 10: Analysis of the system delay across various data sizes

5.3.7 Analysis of the System Delay across Various Computing Capacities of Edge Servers

Fig. 11 depicts the analysis of system delay for various algorithms across varying computing capacities of the edge servers. As can be observed, the increase in the edge server's computing capacity can lower the overall task execution delay because the number of tasks to be executed on the edge server increases with the growth of its processing capacity, thereby reducing the overall system delay. The findings also indicate that the proposed approach surpasses other baseline methods, including DDPG, DQN, DNN, and GA, across different computational capacities. The high computational capacity of the edge servers will be of greater importance during high vehicle density times, as the edge server will serve multiple vehicles. Considering the experimental results under the 15 GHz computing capacity of the edge server as an example, the HetGNN demonstrates superior performance than DDPG, DQN, DNN, and GA, by 22.26%, 30.92%, 28.41%, and 32.91%, respectively.

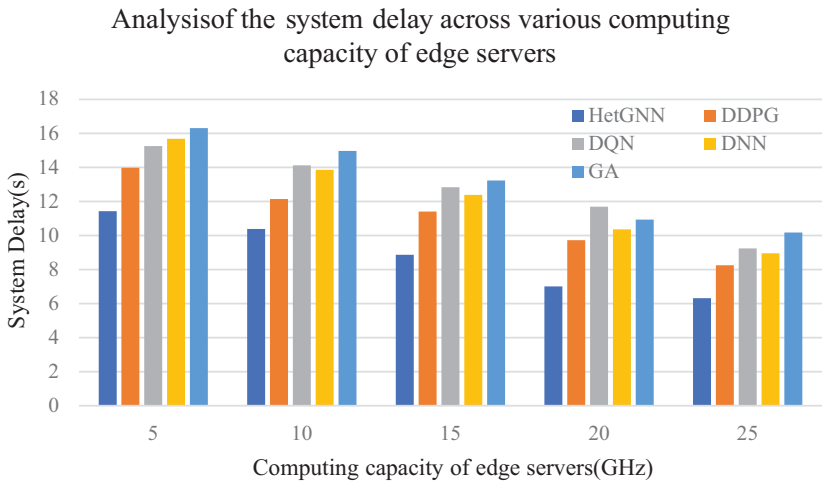

Figure 11: Analysis of the system delay across various computing capacities of edge servers

5.3.8 Analysis of the Inference Time of the Proposed and Baseline Algorithms

Table 6 displays the inference times of the proposed HetGNN alongside other baseline algorithms. This shows that the inference time of the proposed algorithm is comparatively smaller than that of the DDPG, DQN, DNN, and GA algorithms, making it a considerable solution for real-time implementation in the IoV environment.

Table 6: Inference times of algorithms

Algorithm	HetGNN	DDPG	DQN	DNN	GA
Inference time (s)	0.321	0.56	0.89	0.46	1.34

5.3.9 Analysis of the Task Failure Rate

The task failure rate ($\boldsymbol{TFR}$) quantifies the percentage of unsuccessful tasks on all vehicles, which is computed using Eq. (23):

$$\boldsymbol{TFR} = (\mathbf{1} - \boldsymbol{Task_{comp}}/\boldsymbol{Task_{gen}}) * \mathbf{100} \tag{23}$$

where $\boldsymbol{Task_{comp}}$ is the total number of tasks completed on all vehicles and $\boldsymbol{Task_{gen}}$ is the total number of tasks generated on all vehicles. Task failure occurs because of insufficient resources or QoS violations. Table 7 represents the task failure rate of the proposed algorithm in various scenarios.

Fig. 12 illustrates the task failure rate analysis across various scenarios, including variations in the number of vehicles, edge servers, data size, and computational capacity. The results indicate that the proposed HetGNN achieves a lower task failure rate compared to baseline algorithms, demonstrating its scalability and robustness. By modeling the IoV environment as a graph and leveraging an attention mechanism, HetGNN optimally determines task offloading decisions, thereby minimizing the task failure rate.

Table 7: Analysis of the task failure rate

No. of vehicles	25	35	45	55	65
Task failure rate (%)	18	24	29	35	41
No. of edge servers	14	16	18	20	22
Task failure rate (%)	42	35	27	20	14
Task data size (Mb)	10	20	30	40	50
Task failure rate (%)	13	19	22	28	31
Computing capacity of edge servers (GHz)	5	10	15	20	25
Task failure rate (%)	43	36	27	19	15

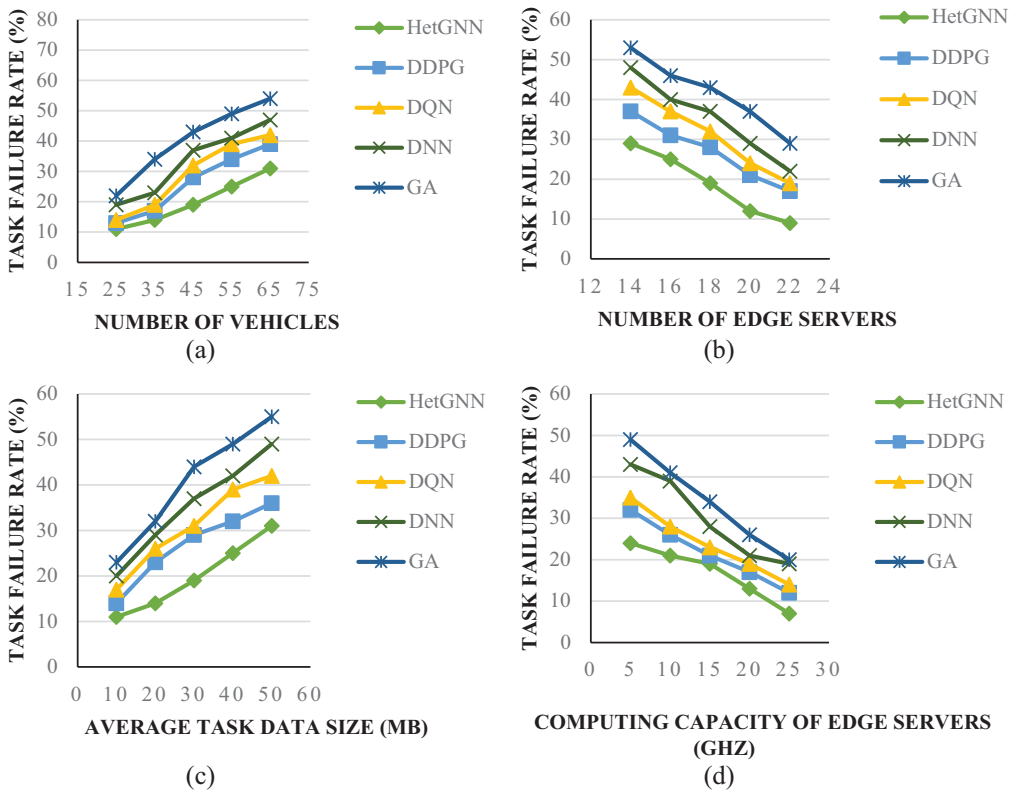

Figure 12: Comparison of task failure rate of the proposed HetGNN with other baseline methods by considering (a) various number of vehicles, (b) various number of edge servers, (c) various task data size, and (d) various computing capacity of edge servers

5.3.10 Analysis of Trade-off between Number of Edge Servers and Resource Utilization

In the Fig. 13, the x-axis represents the number of edge servers, while the y-axis denotes the task acceptance rate. As evident from the figure, the genetic algorithm demonstrates the lowest resource utilization in terms of task acceptance rate. Across varying numbers of edge servers, HetGNN achieves a task acceptance rate between 70% and 90%, indicating efficient utilization of edge server resources and a reduced task failure rate compared to other baseline methods. In contrast, other baseline methods exhibit lower task acceptance rates, ranging from 70% to 80%, even with a higher number of edge servers, highlighting their lower resource utilization efficiency.

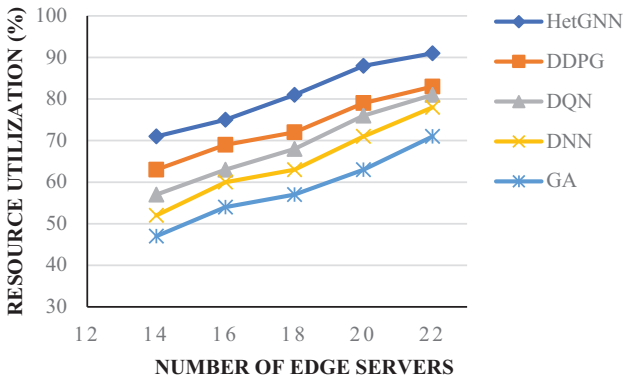

Figure 13: Analysis of trade-off between number of edge servers and resource utilization

5.4 Computational Complexity

The computational complexity with the performance metrics of the proposed HetGNN framework is summarized in the Table 8. The tabulated results compare the proposed framework with other baseline methods in terms of model size, training time, inference time, and system delay. The proposed HetGNN, highlighted in bold, demonstrates competitive computational efficiency with moderate model size, reduced training and inference times, and optimal performance metrics. Notably, the framework achieves a relatively low system delay, outperforming all baseline methods. Regarding network size, the HetGNN framework exhibits a compact memory requirement comparable to DDPG and DQN, signifying a lightweight design that achieves superior performance without excessive computational resource requirements. Additionally, the model's reduced training time and rapid inference capabilities enable efficient decision-making processes. Overall, the results emphasize the proposed framework's superior performance and computational efficiency compared to existing methods.

Table 8: A comparison of the proposed work with various baseline methods considering model size, training time, inference time and system delay

Methods	Model size (KB)	Training time (s)	Inference time (s)	System delay (s)
Genetic algorithm	–	3985.13	1.34	13.23
DNN	13,613	2179.47	0.46	12.39
DQN	24,711	2496.76	0.89	12.84
DDPG	27,319	2678.68	0.56	11.41
HetGNN	**21,131**	**2307.24**	**0.321**	**8.87**

5.5 Ablation Study

In this section, an ablation study is conducted to assess the impact of each component in the IoV environment on task offloading. The attention mechanism is removed from the proposed HetGNN module to analyze its importance for task offloading decisions, and the results are presented in Table 8. The overall system delay and task failure rate increase without an attention mechanism, possibly because of a lack of selective aggregation of adaptive features from the informative nodes rather than aggregating all features from all nodes. The edge server components in the IoV environment, i.e., BSs and UAVs, are excluded, and the results are depicted in Table 9, which shows dramatic increases in the overall system delay and task failure

rate. This shows the importance of MEC in the IoV environment for effective and efficient task offloading since offloading only to the cloud results in a large system delay which is not suitable for vehicular applications as it may have a high task failure rate.

Table 9: Ablation study on the proposed algorithm performance

Model	System delay (s)	Task failure rate (%)
HetGNN	7.67	25
w.o attention	17.43	32
w.o edge servers	44.512	68

5.6 Discussions

The performance evaluation section provides compelling evidence supporting the superior performance of the proposed privacy preserving unsupervised HetGNN task offloading algorithm in comparison to various baseline methods. It can be observed that the proposed HetGNN maintains a lower task failure rate even under high vehicle density and a minimal number of edge servers compared to existing studies [41,42]. This ensures the scalability and robustness of HetGNN to the varying numbers of vehicular users and edge servers. The experimental results further indicate that the proposed graph neural network-based framework shows better results than DRL algorithms compared to the literature [43] in reducing system delay. A comprehensive analysis of the results indicates that the proposed scheme consistently achieves stable performance, even in scenarios with a fluctuating number of vehicles and edge servers. This underscores the adaptability of HetGNNs to the dynamic IoV environment. With the GNN's ability to extract the essential structural features from the graph, the proposed scheme excels in the decision-making process. The HetGNN architecture is designed to adapt to dynamic changes in the number of vehicles and, consequently, the number of nodes by leveraging its training on diverse graph structures. When new vehicular nodes are introduced, HetGNN can accommodate the updated graph topology without retraining, as it generalizes well to varying network configurations. The edge weights for newly added nodes are determined based on the presence of communication links and the transmission rates between the new node and existing nodes. This ensures the model remains robust and responsive to topology changes, maintaining optimal task-offloading decisions in vehicular environments. The Controller can leverage the proposed HetGNN model. In this context, the RSUs, UAVs, and vehicles within its purview serve as nodes in the graph. Employing the HetGNN for analysis enables the identification of optimal task offloading decisions, ultimately reducing the overall task failure rate and ensuring the QoS for vehicular applications. Given the escalating number of connected vehicles, the demands of vehicular applications are on the rise, necessitating effective task offloading solutions. Consequently, the proposed work serves as a viable solution in the current landscape, addressing the increasing requirements of vehicular applications and contributing to the overall enhancement of connected vehicles supporting the IoV system.

6 Conclusion

In this work, the IoV environment is conceptualized as a heterogeneous graph. By employing graph modeling, a comprehensive GNN-based framework is introduced to address the task offloading problem within an IoV environment. We have proposed a lightweight secure unsupervised HetGNN task offloading algorithm to address the scalability and privacy challenges that are caused by a dynamic number of vehicles and malicious servers, respectively. Furthermore, DPSGD is adopted to ensure the privacy of vehicles and

prevent inference attacks. The performance of the proposed method is evaluated with varying parameters. The simulation results demonstrated that the lightweight HetGNN method can efficiently and flexibly optimize the task offloading problem in an IoV scenario with good performance, convergence speed, and generalizability to various inputs. It outperforms all the considered baseline methods, positioning it as an exceptionally promising solution for real-time implementation in the IoV environment with 0.321 s of inference time and a 13% to 19% task failure rate depending on the scalability of the environment.

While our findings highlight the potential of graph attention networks along with differential privacy for scalability, robustness, and privacy preservation, as evidenced by reduced system delay and task failure rates, several limitations remain. One key challenge is incorporating energy efficiency to make the proposed scheme viable for real-time IoV environments. Additionally, the processing time of offloaded tasks could be further optimized through content caching and enhanced resource management strategies. Future work should focus on energy-aware task offloading mechanisms and further delay optimization techniques to improve real-time performance and ensure broader applicability of the proposed framework.

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions, which greatly improved the presentation of this paper.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Aishwarya Rajasekar, Vetriselvi Vetrian; data collection: Aishwarya Rajasekar; analysis and interpretation of results: Aishwarya Rajasekar; draft manuscript preparation: Aishwarya Rajasekar, Vetriselvi Vetrian. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Guo H, Zhou X, Liu J, Zhang Y. Vehicular intelligence in 6G: networking, communications, and computing. Veh Commun. 2022;33:100399. doi:10.1016/j.vehcom.2021.100399.
2. Ahmed M, Raza S, Mirza MA, Aziz A, Khan MA, Khan WU, et al. A survey on vehicular task offloading: classification, issues, and challenges. J King Saud Univ Comput Inf Sci. 2022;34(7):4135–62. doi:10.1016/j.jksuci.2022.05.016.
3. Prakash J, Murali L, Manikandan N, Nagaprasad N, Ramaswamy K. Retraction note: a vehicular network based intelligent transport system for smart cities using machine learning algorithms. Sci Rep. 2024;14(1):27001. doi:10.1038/s41598-024-78141-8.
4. Zhang J, Letaief KB. Mobile edge intelligence and computing for the Internet of vehicles. Proc IEEE. 2020;108(2):246–61. doi:10.1109/JPROC.2019.2947490.
5. Mao B, Tang F, Kawamoto Y, Kato N. Optimizing computation offloading in satellite-UAV-served 6G IoT: a deep learning approach. IEEE Netw. 2021;35(4):102–8. doi:10.1109/MNET.011.2100097.
6. Alseid M, El-Moursy AA, Alfawaz O, Khedr AM. MSSAMTO-IoV: modified sparrow search algorithm for multi-hop task offloading for IoV. J Supercomput. 2023;79(18):20769–89. doi:10.1007/s11227-023-05446-2.
7. Wan N, Luo Y, Zeng G, Zhou X. Minimization of VANET execution time based on joint task offloading and resource allocation. Peer Peer Netw Appl. 2023;16(1):71–86. doi:10.1007/s12083-022-01385-6.

8. Alruwais N, Alabdulkreem E, Kouki F, Aljehane NO, Allafi R, Marzouk R, et al. Farmland fertility algorithm based resource scheduling for makespan optimization in cloud computing environment. Ain Shams Eng J. 2024;15(6):102738. doi:10.1016/j.asej.2024.102738.
9. Xu X, Jiang Q, Zhang P, Cao X, Khosravi MR, Alex LT, et al. Game theory for distributed IoV task offloading with fuzzy neural network in edge computing. IEEE Trans Fuzzy Syst. 2022;30(11):4593–604. doi:10.1109/TFUZZ.2022.3158000.
10. Fan W, Hua M, Zhang Y, Su Y, Li X, Tang B, et al. Game-based task offloading and resource allocation for vehicular edge computing with edge-edge cooperation. IEEE Trans Veh Technol. 2023;72(6):7857–70. doi:10.1109/TVT.2023.3241286.
11. Zhang M. Development of analytical offloading for innovative Internet of vehicles based on mobile edge computing. J Grid Comput. 2023;22(1):4. doi:10.1007/s10723-023-09719-1.
12. Zhang C, Zhang W, Wu Q, Fan P, Fan Q, Wang J, et al. Distributed deep reinforcement learning-based gradient quantization for federated learning enabled vehicle edge computing. IEEE Internet Things J. 2025;12(5):4899–913. doi:10.1109/JIOT.2024.3447036.
13. Wu Q, Wang S, Ge H, Fan P, Fan Q, Ben Letaief K. Delay-sensitive task offloading in vehicular fog computing-assisted platoons. IEEE Trans Netw Serv Manag. 2023;21(2):2012–26. doi:10.1109/TNSM.2023.3322881.
14. Ferrag MA, Maglaras L, Moschoyiannis S, Janicke H. Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study. J Inf Secur Appl. 2020;50(1):102419. doi:10.1016/j.jisa.2019.102419.
15. Liang L, Ye H, Yu G, Li GY. Deep-learning-based wireless resource allocation with application to vehicular networks. Proc IEEE. 2020;108(2):341–56. doi:10.1109/JPROC.2019.2957798.
16. Li J, Luo G, Cheng N, Yuan Q, Wu Z, Gao S, et al. An end-to-end load balancer based on deep learning for vehicular network traffic control. IEEE Internet Things J. 2019;6(1):953–66. doi:10.1109/JIOT.2018.2866435.
17. Elwekeil M, Wang T, Zhang S. Deep learning for environment identification in vehicular networks. IEEE Wirel Commun Lett. 2020;9(5):576–80. doi:10.1109/LWC.2019.2959768.
18. Mehmood A, Khan TA, Muhammad A, Song WC. Multi-class traffic density forecasting in IoV using spatio-temporal graph neural networks. In: 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS); 2022 Sep 28–30; Takamatsu, Japan: IEEE; 2022. p. 1–6.
19. Li R, Qin Y, Wang J, Wang H. AMGB: trajectory prediction using attention-based mechanism GCN-BiLSTM in IOV. Pattern Recognit Lett. 2023;169(7):17–27. doi:10.1016/j.patrec.2023.03.006.
20. Wang C, Wang L, Wei S, Sun Y, Liu B, Yan L. STN-GCN: spatial and temporal normalization graph convolutional neural networks for traffic flow forecasting. Electronics. 2023;12(14):3158. doi:10.3390/electronics12143158.
21. Garg A, Chauhan A, Shambharkar PG. Security threats & attacks in IoV environment: open research issues and challenges. In: 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT); 2022 Aug 11–12; Kannur, India: IEEE; 2022. p. 803–10. doi:10.1109/icicict54557.2022.9917816.
22. Lakhan A, Abed Mohammed M, Garcia-Zapirain B, Nedoma J, Martinek R, Tiwari P, et al. Fully homomorphic enabled secure task offloading and scheduling system for transport applications. IEEE Trans Veh Technol. 2022;71(11):12140–53. doi:10.1109/TVT.2022.3190490.
23. Wei D, Zhang J, Shojafar M, Kumari S, Xi N, Ma J. Privacy-aware multiagent deep reinforcement learning for task offloading in VANET. IEEE Trans Intell Transp Syst. 2023;24(11):13108–22. doi:10.1109/TITS.2022.3202196.
24. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: a review of methods and applications. AI Open. 2020;1(1):57–81. doi:10.1016/j.aiopen.2021.01.001.
25. Deng T, Chen Y, Chen G, Yang M, Du L. Task offloading based on edge collaboration in MEC-enabled IoV networks. J Commun Netw. 2023;25(2):197–207. doi:10.23919/JCN.2023.000004.
26. Mirza MA, Yu J, Raza S, Ahmed M, Asif M, Irshad A, et al. MCLA task offloading framework for 5G-NR-V2X-based heterogeneous VECNs. IEEE Trans Intell Transp Syst. 2023;24(12):14329–46. doi:10.1109/TITS.2023.3292140.
27. Sun Y, Wu J, Wu Y, Chen L, Sun W. Efficient approaches for task offloading in point-of-interest based vehicular fog computing. J Supercomput. 2024;80(5):6285–310. doi:10.1007/s11227-023-05698-y.

28. Fan W, Su Y, Liu J, Li S, Huang W, Wu F, et al. Joint task offloading and resource allocation for vehicular edge computing based on V2I and V2V modes. IEEE Trans Intell Transp Syst. 2023;24(4):4277–92. doi:10.1109/TITS.2022.3230430.
29. He S, Xiong S, Ou Y, Zhang J, Wang J, Huang Y, et al. An overview on the application of graph neural networks in wireless networks. IEEE Open J Commun Soc. 2021;2:2547–65. doi:10.1109/OJCOMS.2021.3128637.
30. Min S, Gao Z, Peng J, Wang L, Qin K, Fang B. STGSN—a spatial-temporal graph neural network framework for time-evolving social networks. Knowl Based Syst. 2021;214(13):106746. doi:10.1016/j.knosys.2021.106746.
31. Pramod Kumar P, Sagar K. Reinforcement learning and neuro-fuzzy GNN-based vertical handover decision on Internet of vehicles. Concurr Comput. 2023;35(12):e7688. doi:10.1002/cpe.7688.
32. Nazzal M, Khreishah A, Lee J, Angizi S, Al-Fuqaha A, Guizani M. Semi-decentralized inference in heterogeneous graph neural networks for traffic demand forecasting: an edge-computing approach. IEEE Trans Veh Technol. 2024;73(12):19400–16. doi:10.1109/TVT.2024.3355971.
33. Sant'Ana da Silva E, Pedrini H, dos Santos AL. Applying graph neural networks to support decision making on collective intelligent transportation systems. IEEE Trans Netw Serv Manag. 2023;20(4):4085–96. doi:10.1109/TNSM.2023.3257993.
34. Chen G, Zhou Y, Zeng Q, Zhang YD. Research on knowledge graph-aware offloading optimization and content acquisition methods for air-ground collaboration in VANETs. IEEE Trans Veh Technol. 2024;73(3):4309–25. doi:10.1109/TVT.2023.3328845.
35. Salami Y, Khajehvand V, Zeinali E. A new secure offloading approach for Internet of vehicles in fog-cloud federation. Sci Rep. 2024;14(1):5576. doi:10.1038/s41598-024-56141-y.
36. Li C, Chen Q, Chen M, Su Z, Ding Y, Lan D, et al. Blockchain enabled task offloading based on edge cooperation in the digital twin vehicular edge network. J Cloud Comput. 2023;12(1):120. doi:10.1186/s13677-023-00496-6.
37. Mao M, Hu T, Zhao W. Reliable task offloading mechanism based on trusted roadside unit service for Internet of vehicles. *Ad Hoc* Netw. 2023;139(1):103045. doi:10.1016/j.adhoc.2022.103045.
38. Gao H, Huang W, Liu T, Yin Y, Li Y. PPO2: location privacy-oriented task offloading to edge computing using reinforcement learning for intelligent autonomous transport systems. IEEE Trans Intell Transp Syst. 2023;24(7):7599–612. doi:10.1109/TITS.2022.3169421.
39. Nasr M, Shokri R, Houmansadr A. Improving deep learning with differential privacy using gradient encoding and denoising. arXiv:2007.11524. 2020.
40. Davody A, Adelani DI, Kleinbauer T, Klakow D. On the effect of normalization layers on differentially private training of deep Neural networks. arXiv:2006.10919. 2020.
41. Trabelsi Z, Ali M, Qayyum T. Fuzzy-based task offloading in Internet of Vehicles (IoV) edge computing for latency-sensitive applications. Internet Things. 2024;28(3):101392. doi:10.1016/j.iot.2024.101392.
42. Ehtisham M, Hassan MU, Al-Awady AA, Ali A, Junaid M, Khan J, et al. Internet of vehicles (IoV)-based task scheduling approach using fuzzy logic technique in fog computing enables vehicular *ad hoc* network (VANET). Sensors. 2024;24(3):874. doi:10.3390/s24030874.
43. Moghaddasi K, Rajabi S, Gharehchopogh FS. Multi-objective secure task offloading strategy for blockchain-enabled IoV-MEC systems: a double deep Q-network approach. IEEE Access. 2024;12:3437–63. doi:10.1109/ACCESS.2023.3348513.