This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

Move-LLM: Multimodal LLM-assisted DRL Framework for On-Demand Service Deployment in 6G Vehicular Networks

Gordon Owusu Boateng, *Member, IEEE*, Amine Kidane Ghebreziabiher, Rabeb Mizouni, Azzam Mourad, *Senior Member, IEEE*, Hadi Otrok, *Senior Member, IEEE*, Jamal Bentahar, and Sami Muhaidat, *Senior Member, IEEE*

***Abstract*—The Sixth-Generation (6G) vehicular network paradigm demands intelligent solutions for dynamic on-demand service deployment. Yet, current approaches are limited by generalized services for specific traffic events, fixed Roadside Unit (RSU) locations, and insufficient context awareness in unimodal data inputs. This paper proposes Move-LLM, a Multimodal Large Language Model (MLLM)-assisted Deep Reinforcement Learning (DRL) framework for context-aware on-demand service deployment in 6G vehicular networks. The framework synergizes three core tasks: Perception (P), Recommendation (R), and Deployment (D). For the P task, raw multimodal (visual, textual) data is fed to a vehicular MLLM (Qwen2-VLM fine-tuned on vehicular events datasets) to enhance its inference via environment perception. For the R task, the inference result is utilized to recommend customized on-demand services capable of tackling the traffic event. Considering the fixed locations of RSUs, we introduce an Onboard Unit (OBU) cluster formation mechanism, where OBUs form clusters as alternative node options for hosting the recommended services. For the D task, we develop an improved Deep Deterministic Policy Gradient (DDPG)-based algorithm with the service recommendations as its partial input state, allowing the agent to select the optimal node and resource allocation strategy to deploy on-demand services in a traffic event. Comprehensive simulation results and analysis reveal that our vehicular MLLM significantly improves P and R task accuracy by about 18.52% and 4.07%, respectively, in terms of Google BLEU (GLEU) score compared with GPT-4o-mini. Moreover, Move-LLM selects the optimal node and allocates optimal resources, achieving at least 11.76% expected utility under the hybrid (OBU clusters + RSUs) deployment mode compared to the RSU-only mode.**

***Index Terms*—Multimodal LLMs, vehicular networks, on-demand service deployment, DDPG, 6G.**

I. Introduction

THE advent of the upcoming Sixth-Generation (6G) mobile network technology heralds a new era of transformative advancements in vehicular networks, characterized by ultra-reliable connectivity and pervasive intelligence [1]. To achieve this, vehicular networks should be able to dynamically deploy on-demand services tailored to the diverse and heterogeneous needs of stakeholders (vehicles, passengers, etc.). On-demand service deployment enables the real-time provisioning of specialized services to address the specific demands of users during traffic incidents or events [2], [3]. Such services in the vehicular network paradigm include emergency medical service, real-time traffic management, and video-on-demand streaming. However, deploying these services presents unique challenges due to the rapidly changing topology of the vehicular network environment and the stringent Quality of Service (QoS) requirements of on-demand services.

Cloud computing has primarily been used to host services on centralized cloud servers, with users granted access on demand [4]. While effective for many applications, this approach introduces unacceptable delays, considering that most vehicular network services are delay-sensitive. As a more pronounced alternative, fog and edge computing paradigms bring service access closer to the users by hosting on-demand services on fog nodes, e.g., vehicular Onboard Units (OBUs) [2] or edge nodes, e.g., Roadside Units (RSUs) [5]. RSUs offer significant advantages in terms of connection stability and reliability, yet their fixed physical locations constrain their ability to maintain service continuity under varying vehicle mobility. Conversely, OBUs provide enhanced flexibility in network coverage due to their mobility. However, they face unstable network connections and limited resources for hosting services.

Consider a crowded city street scenario with heavy traffic, as depicted in Fig. 1. Onboard vehicle and traffic light cameras continuously capture video segments, along with textual metadata (e.g., weather conditions, time of day) recorded by various sensors, to detect stalled vehicle movements and report severe congestion. In this case, a traffic flow management service may be recommended and deployed to redirect traffic. Several existing works have explored rule-based [2], [6] and learning-based approaches [5], [7] to enhance on-demand service deployment in vehicular networks. However, rule-based approaches are limited by their reliance on predefined models and static datasets, which constrain their adaptability to dynamic, large-scale, multimodal data. Although learning-based methods can handle substantial volumes of data, they are often hindered by the black-box nature of their models, leading to poor interpretability for contextual inference and decision-making [8]. Additionally,

G. O. Boateng and A. K. Ghebreziabiher are with the KU 6G Research Center, Department of Computer Science, Khalifa University, Abu Dhabi, UAE.

R. Mizouni and H. Otrok are with the Center for Cyber-Physical Systems (C2PS), Department of Computer Science, Khalifa University, Abu Dhabi, UAE.

A. Mourad is with the KU 6G Research Center, Department of Computer Science, Khalifa University, Abu Dhabi, UAE, and also with the Artificial Intelligence Cyber Systems Research Center, Department of CSM, Lebanese American University, Beirut, Lebanon (Corresponding author, email: azzam.mourad@ku.ac.ae)

J. Bentahar is with the KU 6G Research Center, Department of Computer Science, Khalifa University, Abu Dhabi, UAE, and also with the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, QC, Canada.

S. Muhaidat is with the KU 6G Research Center, Department of Computer and Information Engineering, Khalifa University, Abu Dhabi, UAE, and also with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada.

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

Deep Reinforcement Learning (DRL) agents face challenges in collecting extensive real-world training datasets, leading to sample inefficiency and reduced inference accuracy [9].

Large Language Models (LLMs) have demonstrated exceptional learning capabilities across a wide range of tasks, and vehicular network tasks are not an exception [10]. In particular, Multimodal LLMs (MLLMs) offer several distinct advantages by integrating multiple data modalities, and have been leveraged to improve decision-making in autonomous driving [11]. However, integrating LLMs/MLLMs into 6G vehicular networks poses significant challenges, including model complexity, substantial computational overhead, and extensive resource and energy requirements. Motivated by the aforementioned limitations, this paper introduces **Move-LLM**, a novel **MLLM**-assisted DRL framework for **o**n-demand service deployment in 6G **ve**hicular networks. **Move-LLM** consists of a cloud-based vehicular MLLM (i.e., Qwen2-VLM fine-tuned on vehicular traffic events datasets) that performs **Perception (P)** task for contextual inference and **Recommendation (R)** task for service recommendation, and a DRL agent that performs **Deployment (D)** task for optimal node selection and resource allocation. The main contributions of this paper are summarized as follows:

- We present **Move-LLM**, a vehicular MLLM-assisted DRL framework fine-tuned on multimodal vehicular events datasets to provide accurate, context-aware recommendations for on-demand services that match specific network traffic events. To support the high computation resource requirements of the framework, we host the vehicular MLLM on a centralized cloud infrastructure for seamless fine-tuning and inference-based execution of the **P** and **R** tasks.
- To address the challenges of fixed positions of RSUs, limited resources of a single OBU, and the need for flexibility in service access, we propose to benefit from their respective advantages. Specifically, vehicular OBUs form dynamic clusters as a single movable node to host on-demand services, complementing the immobility of RSU nodes.
- Then, the recommended services from the **R** task execution are fed into an improved DDPG-based algorithm as its partial input state to determine an optimal node among OBU clusters and RSUs to execute the **D** task. The aim is to optimize expected utility, QoS, and Resource Utilization (RU) offerings while accounting for resource constraints.

The rest of the paper is structured as follows: Section II discusses related work, and Section III presents the system model. Section IV formulates the optimization problem and solution. Simulation results and analysis are presented in Section V. Finally, conclusions are drawn in Section VI.

II. Related Work

Cloud computing has primarily been used for the deployment of on-demand services, in which services are hosted on centralized cloud servers and made accessible to users as needed [4], [12]. Ammous *et al.* [12] proposed a cloud-enabled real-time routing scheme for Mobility-on-Demand (MoD) electric vehicles requiring in-route charging. Salahuddin *et al.* [4] proposed an RSU cloud architecture that utilizes Software Defined Networking (SDN) to enable dynamic instantiation, replication, and migration of services in the Internet of Vehicles (IoV). However, centralized cloud servers are limited in flexibility and may struggle to handle the high mobility of vehicles. Moreover, cloud-hosted services incur unacceptable delays and may not support delay-sensitive vehicular services.

Fog and edge computing can bring cloud intelligence closer to the edge, processing user requests on demand while alleviating delays and enabling flexible service provisioning [3], [13]. Some authors [3] proposed an On-Demand Capacity Planning (ODCP) framework for optimizing routing strategies in vehicular fog nodes, aiming to maximize profit and QoS. Sami *et al.* proposed a framework for resource and context-aware deployment of containerized micro-services on on-demand vehicular fogs [2] and Unmanned Aerial Vehicles (UAVs) [13]. OBUs were clustered to form a fleet of volunteer vehicles to host the microservices. However, these works mainly focused on container placement, with little emphasis on service provisioning for specific vehicular traffic events.

Several existing works have proposed Machine Learning (ML) techniques to enhance on-demand service placement and deployment in vehicular networks [5], [7], [14]. Wang *et al.* [14] proposed a fine-grained UAV deployment scheme to meet the real-time communication demands of users using a Convolutional Neural Network (CNN). Predictive approaches have been proposed to study time-aware service demands and network congestion, making informed decisions on where to deploy on-demand services. For instance, Zhang *et al.* [7] presented an ML-enabled framework to predictively deploy UAVs as aerial base stations for on-demand wireless service provisioning at hotspot areas. Huang *et al.* [5] proposed a collaborative on-demand dynamic deployment framework that utilized ARIMA to forecast the number of service requests at each edge cloud and DQN to deploy interacting services. However, RL agents struggle to collect sufficient real-world datasets for training, leading to sample inefficiency and inaccurate model inferences. Additionally, rule-based and learning-based approaches are limited by unimodal data inputs, which are ineffective in capturing the complex patterns and associations in network data.

Unlike traditional rule-based and ML-based algorithms, LLMs excel at processing unstructured data while understanding its context and nuances for multidimensional decision-making [15], [16]. Specifically, LLMs have been applied in vehicular networks for various inference and optimization tasks. Chen *et al.* [17] designed an LLM-driven multi-vehicle dispatching and navigation framework that optimally allocates tasks based on vehicle capabilities and operational contexts. Liu *et al.* [18] proposed an edge computing framework where vehicles perform initial LLM computations locally and offload more intensive tasks to RSUs. MLLMs further enhance the capabilities of vehicular networks by integrating and analyzing diverse data types to improve cooperative driving [19] and ADAS [8]. Liu *et al.* [19] proposed a framework leveraging MLLMs with bird's-eye view representations to optimize spatial relationships between cooperative autonomous vehicles and passenger requests. Hu *et al.* [8] proposed a cloud-edge collaborative architecture leveraging MLLMs to improve environmental understanding and decision-making in IoT networks. A smaller model (CogVLM2) and a larger model (ChatGPT-4o) were deployed at the edge and cloud, respectively, to infer traffic scenarios. Then, a DRL-based method was designed to determine the optimal execution location for each task. To address the computational demand of LLM inference,

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

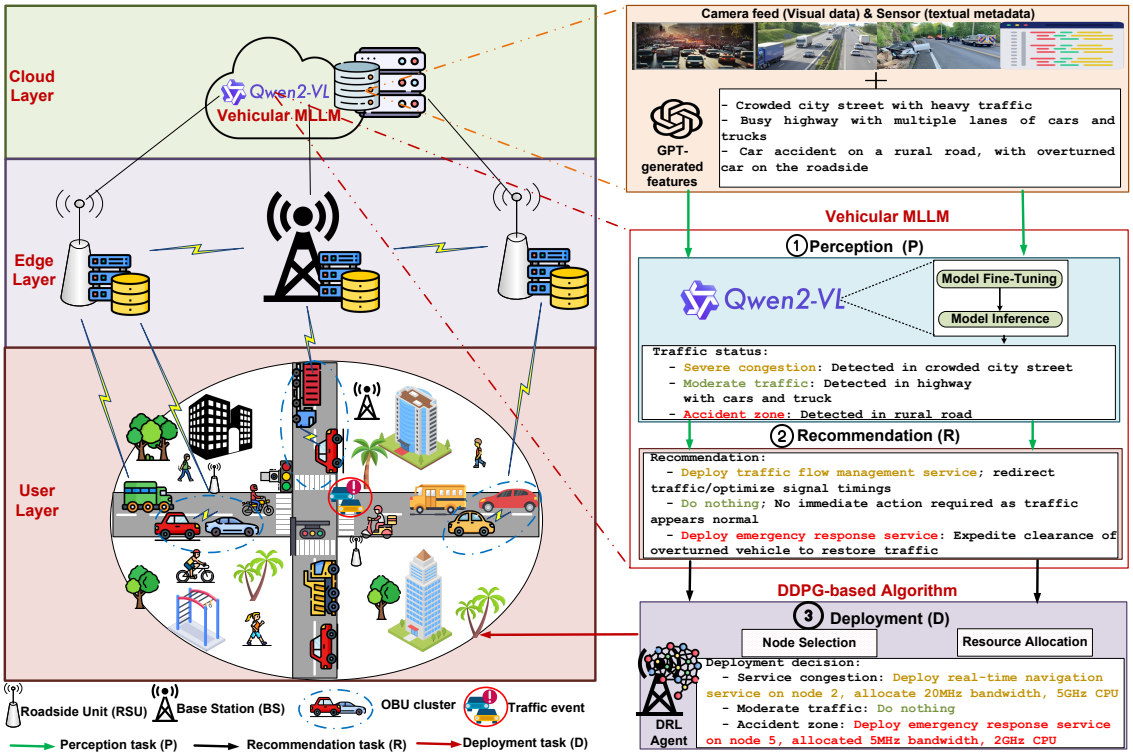


Fig. 1: Proposed system framework.

He *et al.* [20] proposed an active inference approach for LLM inference task offloading and resource allocation in cloud-edge computing. An active inference-based algorithm with rewardless guidance was designed to select the task offloading strategy that maximizes prediction accuracy and minimizes delay.

Despite their significant potential, the complexity of MLLMs, along with their extensive computational and energy requirements, limits their deployment on edge servers. Besides, most MLLMs are trained on unimodal (image or annotated text) data, which limits their ability to understand complex relationships and make meaningful real-time decisions in dynamic vehicular environments. Our work differs from [8] and [20] in terms of application scenario (on-demand service deployment in 6G vehicular networks). Moreover, it extends MLLM inference to service recommendation and service deployment, which are lacking in existing works. Our vehicular MLLM is fine-tuned on unique vehicular event datasets, which distinguishes it from the LLMs in existing work. *To the best of our knowledge, this is the first work to leverage vehicular MLLM-assisted DRL for context-aware environment perception (**P** task), service recommendation (**R** task), and intelligent on-demand service deployment (**D** task) in 6G vehicular networks.*

III. System Model

A. System Framework

Consider a layered and time-variant vehicular network environment with vehicles, pedestrians, and RSUs randomly distributed across the coverage area, as illustrated in Fig. 1. The core components of the system framework and their respective functions are presented as follows:

User layer: The user layer comprises vehicles, pedestrians, and IoT devices. The vehicles are equipped with onboard cameras and OBUs to support visual perception and task computation, while the IoT devices are equipped with cameras and sensors to facilitate real-time environmental perception.

Edge layer: The edge layer serves as a conduit between the user and the cloud layers, comprising RSUs and a BS. The RSUs are equipped with edge servers that provide massive computational and storage resources, as well as cameras and sensors for environmental perception. The BS oversees the vehicular network and matches service requests with optimal hosting nodes.

Cloud layer: The cloud layer consists of a centralized data center with powerful computing and storage resources. It processes high-level raw multimodal data from the user and edge layers for the vehicular MLLM fine-tuning and inference generation [21].

Vehicular MLLM: The vehicular MLLM is hosted in the cloud. It is fine-tuned on multimodal vehicular traffic event datasets to generate inferences for subsequent recommendations and the deployment of on-demand services tailored to specific traffic events.

DRL agent: The DRL agent leverages the service recommendations from the vehicular MLLM to select the optimal node for service hosting, allocating resources to the node for efficient on-demand service deployment.

The detailed operational workflow of the framework is described as follows:

① **Perception (P) task:** Vehicle, RSU, and traffic light cameras capture snapshots of real-time traffic flow and incidents. The sensors collect metadata such as vehicle speed, weather conditions, and traffic density. The multimodal data (camera snapshots and sensor metadata) is uploaded to the cloud for processing and storage. Vehicle-to-infrastructure (V2I) and infrastructure-to-infrastructure (I2I) communication links exist between each vehicle and the cloud, and each RSU/traffic light and the cloud,

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

respectively. To enhance contextual understanding of the multimodal data, each snapshot is analyzed by a GPT model to extract additional features that may not have been provided in the datasets. Then, the multimodal data and GPT-generated features are directly combined into a unified dataset. This unified dataset is used to fine-tune the vehicular MLLM to detect traffic events such as congestion, moderate traffic, and accident zones.

② **Recommendation (R) task:** Based on model inference, the vehicular MLLM recommends accurate and feasible services for on-demand deployment. For instance, if the vehicular MLLM detects an accident on a rural road, it recommends emergency response services, such as clearing overturned vehicles and redirecting traffic. An I2I communication link exists between the cloud and the BS to transfer the service recommendation results to the DRL agent for on-demand service deployment.

③ **Deployment (D) task:** In case of a traffic event, a user initiates a request through the BS to access on-demand services. The BS broadcasts the request to OBU clusters and RSUs. Each OBU cluster and RSU responds with its respective estimated resource allocation, expected QoS, and RU offerings. Then, a DRL agent deployed on the BS executes a node selection and resource allocation algorithm to determine the optimal hosting node based on its offerings and resource constraints.

To meet the 6G millisecond-level latency requirements, **P** and **R** tasks are executed in a long timescale (5-10 seconds) while the **D** task is executed in a short timescale (0.1-1 millisecond). We store all **P** and **R** results of each cycle in a database. When a traffic event occurs, we first look up the database to find a previous deployment strategy that resembles this event (e.g., selecting a comparable event and retrieving its recommended services from an earlier P–R execution). If one exists, we make the deployment decision that best matches the current traffic event. This avoids re-executing the **P** and **R** tasks in each cycle.

B. Vehicular MLLM Preliminaries

The vehicular MLLM is a *fine-tuned Qwen2-VLM*. Like Qwen2-VLM, it basically consists of a vision encoder $g_\psi(\cdot)$, an LLM $f_\Phi(\cdot)$, and a merger $p_\sigma(\cdot)$ connecting the vision encoder and the LLM. The visual data from the vehicular event datasets is processed by the vision encoder into textual output, while the textual metadata is processed by the LLM. Then, the merger aggregates the textual outputs of both the vision encoder and the LLM for inference generation.

The textual data X_{txt} and visual data X_{vis} are embedded into dense feature vectors with modality-specific embeddings $E = E_{txt} + E_{vis}$ and positional encodings $PE = PE_{txt} + PE_{vis}$, where X_{txt} and X_{vis} are encoded into text features $Z_{txt} = f(X_{txt})$ and visual features $Z_{vis} = g(X_{vis})$, respectively. To model dependencies between tokens, the vehicular MLLM employs the self-attention mechanism to compute the query Q, key K, and value V matrices as

$$Q = (E+PE)\times W^Q; K = (E+PE)\times W^K; V = (E+PE)\times W^V, \quad (1)$$

where W^Q, W^K, and W^V are learned weight matrices. Following [22], the attention mechanism computes relationships between tokens as

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2)$$

where d_k is the dimensionality of K. The vehicular MLLM uses multi-head self-attention to capture information from multiple subspaces as

$$MultiHead(Q,K,V) = Concat[head_1;\ldots;head_h]W^O, \quad (3)$$

where $head_h = Attention(QW_h^Q, KW_h^K, VW_h^V)$, and W^O is the learned weight matrix for combining the outputs of the attention heads. The resulting self-attention layer output after multi-head self-attention calculation is given by

$$\mathcal{Z} = MultiHead(Q,K,V), \quad (4)$$

where $\mathcal{Z} = \mathcal{Z}_{txt}, \mathcal{Z}_{vis}$. Then, $\mathcal{Z}_{vis}$ is projected onto a shared embedding space for textual representation using the merger $p_\sigma(\cdot)$, which yields modality-specific embeddings as $H_{vis} = p(\mathcal{Z}_{vis})$, where H_{vis} is the textual representation of $\mathcal{Z}_{vis}$. These embeddings, together with $\mathcal{Z}_{txt}$, are passed through a Multi-Layer Perceptron (MLP) in $f_\Phi(\cdot)$, comprising stacked transformer layers of self-attention and Feed-Forward Networks (FFNs). The final representation $\hat{\mathcal{Z}}$ is used to make event-specific service recommendation Rec_s and an accompanying justification $Justify_s$, expressed as

$$\text{Vehicular MLLM}(\hat{\mathcal{Z}}) = [Rec_s, Justify_s]. \quad (5)$$

Rec_s is a formatted JSON output with textual and numerical vectors, and $Justify_s$ is a textual vector that explains the reason for recommending the set of services for the specific traffic event. Rec_s follows a structured instruction format. If this instruction is not followed by the vehicular MLLM, it is reprompted with a follow-up instruction to adjust its output.

C. Network Model

Let $\mathcal{S} \triangleq \{s = 1, 2, ..., S\}$ denote a set of available services for on-demand deployment. The network comprises a BS, a set of RSUs, and OBUs. The RSU and OBU sets are denoted by $\mathcal{I} \triangleq \{i = 1, 2, ..., I\}$, and $\mathcal{B} \triangleq \{b = 1, 2, ..., B\}$, respectively. A number of OBUs can form a cluster j to be in contention for hosting services, i.e., $\mathcal{J} \triangleq \{j = 1, 2, ..., J\} \subseteq \mathcal{B}$, where $\mathcal{J}$ represents a set of OBU clusters. The system operates over T finite timeslots, defined by the set $\mathcal{T} \triangleq \{t = 1, 2, ..., T\}$. For simplicity, both RSU i and OBU cluster j are collectively referred to as node n, i.e., $n \in \{i, j\}$, unless otherwise specified.

For on-demand service deployment, a requested service req_s can be expressed as

$$req_s = \left[\sum_{s\in\mathcal{S}} s : \phi_s(r_s^{min}, \tau_s^{max})\right], \quad (6)$$

where ϕ_s, r_s^{min}, and τ_s^{max} are the QoS, minimum data rate, and maximum delay requirements of service s, respectively. The BS broadcasts req_s to RSUs and OBU clusters. Each RSU and OBU cluster responds to req_s with $resp_{n,s}$, which is expressed as

$$resp_{n,s} = [n : \rho_{n,s}, \phi_{n,s}(r_{n,s}, \tau_{n,s}), \Psi_{n,s}], \quad (7)$$

where $\rho_{n,s}$, $\phi_{n,s}(r_{n,s}, \tau_{n,s})$, and $\Psi_{n,s}$ denote the estimated resource allocations, expected QoS offerings, and related RU, respectively, for service s. Based on req_s and $resp_{n,s}$, a DRL agent deployed on the BS executes a node selection and resource allocation algorithm to designate a node that optimizes $\rho_{n,s}$, $\phi_{n,s}$, and $\Psi_{n,s}$, considering resource limitations. The total resource

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

owned by node n includes bandwidth $\boldsymbol{W}_n$ (in Hz) and computation $\boldsymbol{C}_n$ (in CPU cycles). Therefore, the fraction of resource required to host service s is given by $\rho_{n,s} = \{\boldsymbol{w}_{n,s}, \boldsymbol{c}_{n,s}\}$. We define a binary node selection variable $x_{n,s}$ as

$$x_{n,s} = \begin{cases} 1, & \text{if} \quad \text{node } n \text{ is selected to host service } s \\ 0, & \text{otherwise} \end{cases}. \quad (8)$$

D. Deployment Model

1) Communication Model: According to the Shannon capacity theory [23], the data rate $r_{n,s}$ that node n can offer to host service s is given by

$$r_{n,s} = \boldsymbol{w}_{n,s} \cdot \log_2 \left(1 + \frac{p_n \cdot g_n}{\sigma^2}\right), \quad (9)$$

where $\frac{p_n \cdot g_n}{\sigma^2}$ denotes the Signal-to-Noise-Ratio (SNR), p_n is the transmission power, g_n is the channel gain, and σ^2 is the variance of Additive White Gaussian Noise (AWGN). The data rate constraint is satisfied if $r_{n,s} \geq r_s^{min}$.

2) Computation Model: For hosting service s on node n, two delay components are considered: transmission delay $\tau_{n,s}^{trans}$ and computation delay $\tau_{n,s}^{comp}$, which can be computed as

$$\tau_{n,s}^{trans} = \frac{l_s}{r_{n,s}}, \qquad \tau_{n,s}^{comp} = \frac{\varsigma_s}{\boldsymbol{c}_{n,s}}, \quad (10)$$

where l_s is the data size (in MB) and ς_s is the CPU cycles (in Megacycles) [24]. We note that OBU cluster formation incurs a significant delay τ_j, which should be accounted for. Therefore, the overall deployment delay is expressed as

$$\tau_{n,s} = \begin{cases} \tau_{n,s}^{trans} + \tau_{n,s}^{comp}, & \text{if} \quad n = i, \\ \tau_{n,s}^{trans} + \tau_{n,s}^{comp} + \tau_j, & \text{if} \quad n = j \end{cases}. \quad (11)$$

The delay constraint is satisfied if $\tau_{n,s} \leq \tau_s^{max}$.

E. Utility Model

Based on $r_{n,s}$ and $\tau_{n,s}$, $\phi_{n,s}$ is computed by [25]

$$\begin{aligned} \phi_{n,s}^{r_{n,s}} &= \frac{1}{1 + e^{-\eta(r_{n,s} - r_s^{min})}}, \\ \phi_{n,s}^{\tau_{n,s}} &= \frac{1}{1 + e^{-\eta(\tau_s^{max} - \tau_{n,s})}}, \end{aligned} \quad (12)$$

where $\phi_{n,s} = \frac{\phi_{n,s}^{r_{n,s}} + \phi_{n,s}^{\tau_{n,s}}}{2}$, in the range $[0, 1]$, and η is a curve-fitting parameter. Given the fraction of resources required to host service s, RU $\Psi_{n,s}$ is defined as the ratio of allocated resources to total resources, and is expressed as

$$\Psi_{n,s} = \frac{1}{2}\left(\frac{\boldsymbol{w}_{n,s}}{\boldsymbol{W}_n} + \frac{\boldsymbol{c}_{n,s}}{\boldsymbol{C}_n}\right), \quad (13)$$

in the range $[0, 1]$.

From (8), (12), and (13), the expected utility $\mathcal{U}_{n,s}$ of node n is given by

$$\mathcal{U}_{n,s} = \sum_{s \in \mathcal{S}} x_{n,s}\Big(\alpha_1 \cdot \phi_{n,s} - \alpha_2 \cdot \Psi_{n,s}\Big), \quad (14)$$

where $\alpha_1 + \alpha_2 = 1$; α_1 and α_2 are coefficients that denote the importance of the utility terms.

Algorithm 1 OBU Cluster Formation and CH Selection

Input: Event location $\mathbf{y}_e(t)$, eligibility score threshold ξ^{thr}
Output: Cluster j and selected CH CH_j

```
Broadcast y_e(t) from BS to all B vehicles in event vicinity
Initialize cluster j ← {}
for each vehicle b in event vicinity do
    Compute eligibility score ξ_b using d_{b,e}, v_b, ρ_b, t_b^{serv}
    if ξ_b ≥ ξ^{thr} then
        Add vehicle b to cluster j
    end if
end for
Rank vehicles in j by ξ_b in descending order
Select top-ranked vehicle as CH_j
while unclustered vehicle b' requests to join j do
    Compute eligibility score ξ_{b'}
    if ξ_{b'} ≥ ξ^{thr} then
        Add b' to cluster j
    end if
end while
If CH_j becomes unavailable, re-elect a new CH via 9-10
```

IV. Problem Formulation and Algorithm

A. OBU Cluster Formation and Cluster Head (CH) Selection

OBU cluster formation serves as a critical mechanism for enabling collaborative service hosting, addressing the challenges posed by fixed RSU positions and the limited resources of a single OBU. While this subsection outlines the foundational steps of OBU cluster formation, a detailed theoretical analysis of intra-cluster communication and CH selection (also referred to as master node election) has been addressed in our prior works [2], [13].

1) OBU cluster formation: When a traffic event occurs, a group of vehicles (also referred to as OBUs, i.e., vehicle and OBU are denoted by b) in close proximity dynamically forms a cluster as a moving node to collaboratively host on-demand services. The process begins with the BS broadcasting the event location coordinate $\mathbf{y}_e(t)$ to all vehicles in the vicinity. Initially, all potential vehicles to join cluster j are disjoint, and j is an empty set $\{\}$. Each vehicle b calculates an eligibility score ξ_b to join the cluster based on its proximity to the event $d_{b,e}$, speed v_b, available resource allocations ρ_b, and the expected serving time t_b^{serv}. Vehicles with ξ_b exceeding a threshold ξ^{thr} ($\xi_b \geq \xi^{thr}$) form an initial cluster j and await CH selection.

2) OBU CH selection: Once the OBU cluster j is formed, a CH CH_j is selected to coordinate the cluster's activities, including communication with the BS and service hosting. Vehicles in j are ranked in descending order according to their eligibility scores ξ_b, and the vehicle with the highest score is selected as CH_j. After initial cluster formation and CH selection, unclustered vehicles in the event's vicinity may request to join the cluster. For each unclustered requesting vehicle b', its eligibility score $\xi_{b'}$ is computed. If $\xi_{b'} \geq \xi^{thr}$, vehicle b' is admitted into cluster j. If CH_j becomes unavailable due to mobility or resource depletion, a new CH is re-elected by reranking the remaining vehicles in the cluster using the eligibility score ξ_b. **Algorithm 1** presents the OBU cluster formation and CH selection procedure.

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

B. Optimization Problem

Following the service recommendation result from (5), the BS selects an optimal hosting node among RSUs and OBU clusters, considering $\rho_{n,s}$, $\phi_{n,s}$, and $\Psi_{n,s}$ of each node, as well as resource constraints. Then, the selected node allocates optimal resources required to host the services that cater for the traffic event. We formulate the node selection and resource allocation problem as a Mixed Integer Non-Linear Programming (MINLP) problem, with the objective of selecting an optimal hosting node that maximizes the utility $\mathcal{U}_{n,s}$. Therefore, the optimization problem is expressed as

$$\begin{aligned} \underset{\boldsymbol{w}_{n,s},\boldsymbol{c}_{n,s}}{\text{maximize}} \quad & \mathcal{U}_{n,s} \\ \text{s. t.:} \quad & \textbf{C1}\text{: } x_{n,s} \in \{0,1\} \quad \forall n \in \mathcal{N}, s \in \mathcal{S}, \\ & \textbf{C2}\text{: } \boldsymbol{w}_{n,s} \leq \boldsymbol{W}_n,\ \boldsymbol{c}_{n,s} \leq \boldsymbol{C}_n, \\ & \textbf{C3}\text{: } \rho_{n,s} \geq 0, \\ & \textbf{C4}\text{: } r_{n,s} \geq r_s^{min},\ \tau_{n,s} \leq \tau_{n,s}^{max}, \\ & \textbf{C5}\text{: } \sum_{s \in \mathcal{S}} x_{n,s} \leq \Gamma_n. \end{aligned} \tag{15}$$

Constraint **C1** ensures that node n is selected to host at least one service s. Constraint **C2** satisfies the bandwidth and computation constraints. Constraint **C3** ensures that the resource allocations of node n cannot be less than 0 (otherwise, there is no optimization problem). Constraint **C4** satisfies the data rate and delay constraints. Finally, Constraint **C5** indicates that the number of services node n can host should not exceed its service hosting capacity Γ_n. The outcome of (15) is the optimal node n^* selected to host service s and its corresponding optimal resource allocations $\rho_{n,s}^*$, i.e., $\mathcal{U}_{n,s}^* = (n^*, \rho_{n,s}^*)$.

The optimization problem is an MINLP problem with binary integer variable $x_{n,s}$ and continuous variable $\rho_{n,s}$. Given these characteristics, it is reasonable to conclude that (15) is NP-hard [26], [27]. Given the dynamics in the vehicular network environment and long-term optimization goals, conventional optimization methods cannot effectively address this intricate problem [28]. DRL techniques have shown great promise for learning from experience and dynamically resolving complex decision-making problems under varying network conditions without a prespecified model or complete network information [29]. Therefore, we transform the optimization problem in (15) into a stochastic Partially-Observable Markov Decision Process (POMDP) and propose an improved DDPG-based algorithm with the recommendation output from the vehicular MLLLM as its partial input space to determine n^* and $\rho_{n,s}^*$.

C. Improved DDPG-based Algorithm

The stochastic POMDP can be defined as the tuple $\langle \vartheta, \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R}, \boldsymbol{S}' \rangle$, where ϑ denotes the agent, $\boldsymbol{S}$ denotes the finite state space set, $\boldsymbol{A}$ denotes the action space set, $\boldsymbol{R}$ denotes the reward function set, and $\boldsymbol{S}'$ denotes the next state space set. In POMDP, agent ϑ partially perceives an observation $\boldsymbol{o}_\vartheta^t$ in state $\boldsymbol{s}_\vartheta^t \in \boldsymbol{S}$, selects action $\boldsymbol{a}_\vartheta^t \in \boldsymbol{A}$, observes reward $\boldsymbol{r}_\vartheta^t \in \boldsymbol{R}$, and transitions into next state $\boldsymbol{s}'_\vartheta \in \boldsymbol{S}'$. The goal of agent ϑ is to find an optimal strategy $\pi(\boldsymbol{s}/\boldsymbol{A})$ that maps state $\boldsymbol{s}$ to action $\boldsymbol{a}$ in a set of possible actions $\boldsymbol{A}$, thereby maximizing its expected cumulative reward, and is given by

$$\mathcal{V}_\pi(\boldsymbol{s}^t) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \cdot \boldsymbol{r}^t | \boldsymbol{s}_0 = \boldsymbol{s} \right], \tag{16}$$

where $\mathcal{V}_\pi(\boldsymbol{s}^t)$ is the state-value function, $\gamma \in [0,1]$ is the discount factor, and $\boldsymbol{s}_0$ is the initial state. Then, the state-action value function (Q-function) is defined as

$$\mathcal{Q}_\pi(\boldsymbol{s}^t, \boldsymbol{a}^t) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \cdot \boldsymbol{r}^t | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a} \right]. \tag{17}$$

We design an improved DDPG-based algorithm to learn optimal node selection and resource allocation strategies that maximize expected utility. The state space, action space, and reward function of the POMDP can be customized for the optimization problem as follows:

*State space (***s***)*: At timeslot t, agent ϑ perceives the state $\boldsymbol{s}_\vartheta^t$ from the vehicular network environment. Rec_s from the vehicular MLLM serves as a partial input state to agent ϑ. We omit $Justify_s$ from the DRL state space definition, as its sole purpose is to self-guardrail the vehicular MLLM to recommend accurate and feasible services from the service pool to tackle the specific traffic event. The DRL state space is therefore expressed as

$$\boldsymbol{s}_\vartheta^t \triangleq \left\{ Rec_s, req_s, resp_{n,s}, \mathcal{N}^{avail} \right\}, \tag{18}$$

where $\mathcal{N}^{avail} = [n_1^{avail}, n_2^{avail}, ..., n_N^{avail}]$ is the list of available nodes. We assume that each available node n^{avail} is capable of fully responding to req_s. It is noteworthy that $(Rec_s|\boldsymbol{s}_\vartheta^t)$ is a simplified output from the vehicular MLLM's **R** task after extensive fine-tuning on raw multimodal input data. With this simplification, Rec_s reduces the state space, thereby lowering the algorithm's computational complexity and improving decision-making efficiency.

*Action space (***a***)*: Given the state $\boldsymbol{s}_\vartheta^t$, agent ϑ selects action $\boldsymbol{a}_\vartheta^t$ from a set of possible actions $\boldsymbol{A}$. Note that agent ϑ selects actions based solely on the numerical component of Rec_s and other terms in $\boldsymbol{s}_\vartheta^t$. For instance, agent ϑ may assign "emergency medical services" a service ID 3 and "Accident video reporting" a service ID 1. Each service ID is mapped to the corresponding resource and QoS requirements of the service. The action space is expressed as

$$\boldsymbol{a}_\vartheta^t \triangleq \{n^*, \rho_{n,s}^*\}, \tag{19}$$

where n^* is the selected optimal node, and $\rho_{n,s}^* = \{\boldsymbol{w}_{n,s}^*, \boldsymbol{c}_{n,s}^*\}$ is the optimal resources allocated for service deployment.

*Reward function (***r***)*: Agent ϑ obtains reward $\boldsymbol{r}_\vartheta^t$ for taking action $\boldsymbol{a}_\vartheta^t$ in state $\boldsymbol{s}_\vartheta^t$, which is the expected utility defined in (14), and is expressed as

$$\boldsymbol{r}_\vartheta^t \triangleq \mathcal{U}_\vartheta^t = \sum_{s \in \mathcal{S}} x_{n,s} \left(\alpha_1 \cdot \phi_{n,s}^t - \alpha_2 \cdot \Psi_{n,s}^t \right). \tag{20}$$

Agent ϑ receives a penalty if the resource and QoS constraints are violated.

Algorithm Design: From (19), it can be observed that $\boldsymbol{a}_\vartheta^t$ consists of both discrete action n^* and continuous action $\rho_{n,s}^*$. We can either discretize $\rho_{n,s}^*$ (for DRL methods such as DQN) or relax n^* into a continuous action space (for DRL methods such as DDPG). However, adapting continuous action spaces for DQN results in the curse of dimensionality, while naive discretization may unnecessarily disrupt the relevant information in the action space. Therefore, we choose DDPG as it operates over continuous action spaces [30]. One common way to relax n^* is to convert it from $x_{n,s} \in \{0,1\}$ to $\hat{x}_{n,s} \in [0,1]$. During training, $\hat{x}_{n,s}$ takes

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

values in the continuous range $[0, 1]$. After the agent obtains $\hat{x}_{n,s}$, we apply a threshold thr to map the value back to binary as

$$x_{n,s} = \begin{cases} 1, & \text{if} \quad \hat{x}_{n,s} \geq thr \\ 0, & \text{otherwise} \end{cases}. \tag{21}$$

DDPG is an actor-critic method that concurrently learns a Q-function and a policy function. Its structure includes the primary network and target network, each for the actor and critic. The primary actor and critic networks are defined as $\pi(\boldsymbol{s}; \theta^{\pi})$ and $\mathcal{Q}(\boldsymbol{s}, \boldsymbol{a}; \theta^{\mathcal{Q}})$, respectively. The actor function $\pi(\boldsymbol{s}; \theta^{\pi})$ is maintained by a parameter θ^{π} that specifies the current policy by deterministically mapping states to a specific action, while the critic function $\theta^{\mathcal{Q}}$ is learned using the Bellman equation from Q-learning as

$$\boldsymbol{J}(\theta) = \mathbb{E}_{\boldsymbol{s} \sim \mathcal{D}}[\theta^{\mathcal{Q}}(\boldsymbol{s}, \theta^{\pi}(\boldsymbol{s}))]. \tag{22}$$

At timeslot t, the actor chooses an action $\boldsymbol{a}^t$ based on current state $\boldsymbol{s}^t$ and current policy π as

$$\boldsymbol{a}^t = \pi(\boldsymbol{s}^t, \theta^{\pi}) + \boldsymbol{N}^t, \tag{23}$$

where $\boldsymbol{N}^t$ is the Ornstein-Uhlenbeck (OU) noise. The critic network evaluates the Q-values of the actor network using the gradient descent method [31]. Based on the Bellman equation, the Q-value can be computed by

$$\mathcal{Q}^{\pi}(\boldsymbol{s}^t, \boldsymbol{a}^t) = \mathbb{E}[\boldsymbol{r}(\boldsymbol{s}^t, \boldsymbol{a}^t) + \gamma \cdot \mathcal{Q}^{\pi}(\boldsymbol{s}', \pi(\boldsymbol{s}', \theta^{\mathcal{Q}}))]. \tag{24}$$

Then, the target critic network calculates the target Q-value as

$$y^t = \boldsymbol{r}^t + \gamma \cdot \mathcal{Q}'(\boldsymbol{s}', \pi', (\boldsymbol{s}|\theta^{\pi'}), \theta^{\mathcal{Q}'})). \tag{25}$$

The critic network can be updated by minimizing the Bellman loss as

$$\mathcal{L}_{\vartheta}(\mathcal{Q}_{\vartheta}) = \mathbb{E}_{(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}') \sim \mathcal{D}}[(y_{\vartheta} - \mathcal{Q}_{\vartheta}(\boldsymbol{s}, \boldsymbol{a}; \theta))^2], \tag{26}$$

where $y_{\vartheta} = \boldsymbol{r} + \gamma \cdot \mathcal{Q}(\boldsymbol{s}', \boldsymbol{a}', \theta^{\mathcal{Q}})$. The policy gradient of the DPG objective function with respect to θ^{π} is given by

$$\bigtriangledown \theta^{\pi} J(\pi) = \mathbb{E}_{\boldsymbol{s}, \boldsymbol{a} \sim \mathcal{D}}[\bigtriangledown \theta^{\pi} \pi(\boldsymbol{s}|\theta^{\pi})|_{\boldsymbol{s}=\boldsymbol{s}^t} \bigtriangledown \theta^{\mathcal{Q}}(\boldsymbol{s}, \boldsymbol{a}|\theta^{\mathcal{Q}})|_{\boldsymbol{s}=\boldsymbol{s}^t, \boldsymbol{a}=\pi(\boldsymbol{s}^t)}]. \tag{27}$$

Finally, the target network is updated using soft updates as

$$\begin{aligned} \theta^{\pi'} &\leftarrow \mu \theta^{\pi} + (1 - \mu) \theta^{\pi'} \\ \theta^{\mathcal{Q}'} &\leftarrow \mu \theta^{\mathcal{Q}} + (1 - \mu) \theta^{\mathcal{Q}'}, \end{aligned} \tag{28}$$

where μ denotes the soft update factor. The DDPG framework is shown in Fig. 2.

The computational complexity of **Algorithm 2** is as follows: the complexity of the forward and backward pass each for the actor network and critic network is $\mathcal{O}(\mathbb{L}\mathbb{H}^2)$, where $\mathbb{L}$ and $\mathbb{H}$ are the number of layers in the neural network and the number of neurons per layer, respectively. Sampling mini-batch $\boldsymbol{M}'$ from replay memory $\boldsymbol{M}$ takes complexity $\mathcal{O}(\mathbb{M})$. The primary actor and critic network update using gradient descent contributes to additional complexity of $\mathcal{O}(\mathbb{M} \cdot \mathbb{L}\mathbb{H}^2)$. Updating the target networks involves copying weights with a soft update factor μ, with added complexity of $\mathcal{O}(\mathbb{L}\mathbb{H})$. If **Algorithm 2** executes for $\hat{T}$ decision epochs, its complexity is $\mathcal{O}(\hat{T} \cdot \mathbb{M} \cdot \mathbb{L}\mathbb{H}^2)$. The complexity of **Algorithm 1** is given by $\mathcal{O}(BlogB)$. Considering that **Algorithm 1** executes in **Algorithm 2**, the overall computational complexity of **Algorithm 2** is $\mathcal{O}(\hat{T} \cdot (\mathbb{M} \cdot \mathbb{L}\mathbb{H}^2 + BlogB))$.

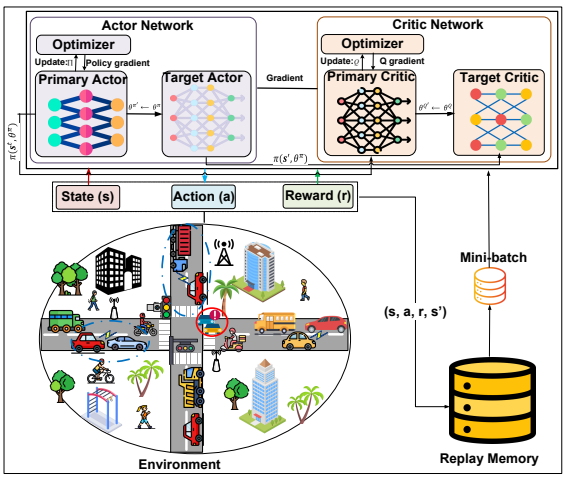


Fig. 2: DDPG framework.

Algorithm 2 Improved DDPG-based Algorithm

Input: Initial actor network $\pi(\boldsymbol{s}; \theta^{\pi})$ and critic network $\mathcal{Q}(\boldsymbol{s}, \boldsymbol{a}; \theta^{\mathcal{Q}})$ with weights θ^{π} and $\theta^{\mathcal{Q}}$, respectively
Output: Optimal node n^* and resource allocation $\rho^*_{n,s}$

```
Initialize target actor and critic networks with weights θ^π' ← θ^π and θ^Q' ← θ^Q, replay memory M and minibatch M'
for each episode do
    Set up the simulation environment
    Execute Algorithm 1
    for each decision epoch t do
        Obtain state s^t
        Select action a^t for exploration according to (23)
        Execute action a^t, calculate reward r^t and next state s'
        Store experience tuple ⟨s^t, a^t, r^t, s'⟩ in M
        Sample random mini-batch of transitions ⟨s^v, a^v, r^v, s'⟩ from M
        Compute target value y^t using (25)
        Update critic by minimizing loss L_ϑ(Q_ϑ) using (26)
        Update actor policy by ▽θ^π J(π) using (27)
        Update target networks by soft update using (28)
    end for
end for
```

V. Performance Evaluation

In this section, we evaluate the performance of the proposed **Move-LLM** framework through extensive, comprehensive simulations. The evaluation is conducted in two folds: 1) vehicular MLLM performance evaluation in terms of **P** and **R** task execution; 2) **Move-LLM** performance evaluation in terms of end-to-end task execution, with focus on **D** task execution.

A. Vehicular MLLM Performance Evaluation

We fine-tune *Qwen2-VLM* on raw multimodal vehicular event datasets to obtain our vehicular MLLM. Then, it is used to perform **P** and **R** tasks. Our choice of *Qwen2-VLM* is due to its ability to process multimodal inputs of tokenized textual and visual data. We utilize Qwen2 [32] as the LLM with 2 billion (2B) or 7B parameters. The vision encoder is based on the Vision Transformer (ViT), with approximately 675 million parameters. For simplicity in naming, we refer to our vehicular

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

MLLM as "Qwen-FT" in the following. Two variants of Qwen are deployed: a 2B-parameter variant (Qwen-2B-FT) and a 7B-parameter variant (Qwen-7B-FT).

1) Dataset Description: We evaluate Qwen-FT using the following open-source datasets:

Highway accident detection and classification dataset [33]: This dataset contains various highway scenarios data, including intersections and exits, with labeled information on accident types. It contains 2780 video segments, each lasting 3-8 seconds. The video segments are categorized into 11 incident classes and 1 class for normal (moderate) traffic. The list of incidents includes collision with a motorcycle, collision with a stationary object, drifting or skidding, fire or explosion, head-on collision, objects falling, pedestrian hit, rear-end collision, rollover, side collision, and other crashes, which form the textual metadata.

Highway traffic videos dataset [34]: This dataset consists of highway surveillance video footage containing 262 video segments, each lasting about 5 seconds. Each video includes traffic conditions (light, medium, heavy), weather conditions (overcast, rainy, clear), time conditions (daytime, nighttime), and a timestamp, which constitute the textual metadata.

The combined set of video segments is denoted as $\mathbf{V}$, and the original features in both datasets are referred to as *High-Level features* ($\mathbf{F_{HL}}$). To enable Qwen-FT well identify correlations between $\mathbf{V}$ and $\mathbf{F_{HL}}$, we feed each video segment in $\mathbf{V}$ into ChatGPT-4o to artificially extract environment information that may not have been explicitly provided in the datasets and obtain the corresponding textual metadata, which describes the main observations in the video segment. These extracted GPT-assisted features provide the expected inference from Qwen-FT, including the expected severity, a context summary, and the most probable cause of the traffic event, and are referred to as *Low-Level features* ($\mathbf{F_{LL}}$). With $\mathbf{F_{HL}}$ and $\mathbf{F_{LL}}$, we are able to complement any data collection errors before fine-tuning Qwen-FT with the datasets.

We classify $\mathbf{V}$, $\mathbf{F_{HL}}$, and $\mathbf{F_{LL}}$ into two sets of data, one for executing the **P** task and another for the **R** task as follows:

- **Dataset 1 (D1): D1** comprises $\mathbf{V}$, $\mathbf{F_{HL}}$, and $\mathbf{F_{LL}}$, and is used for **P** task execution. $\mathbf{V}$ and $\mathbf{F_{HL}}$ are the input and $\mathbf{F_{LL}}$ is the expected output.
- **Dataset 2 (D2): D2** comprises $\mathbf{F_{LL}}$ and $[Rec_s, Justify_s]$, and is used for **R** task execution. $\mathbf{F_{LL}}$ is the input, and $[Rec_s, Justify_s]$ is the output.

2) Comparative MLLMs: We compare Qwen with pre-trained proprietary vehicle-specific MLLMs, fine-tuned vehicle-specific MLLMs, and an existing state-of-the-art vehicle-specific MLLM as follows:

- **Pre-trained GPT-4o-mini (GPT-4o-mini) [35]:** This is a pre-trained proprietary vehicle-specific GPT-4o-mini MLLM.
- **Pre-trained GPT-4o+CogVLM2 (GPTCOG) [8]:** This is a pre-trained version of the state-of-the-art vehicle-specific MLLM implemented in [8]. It is a combination of pre-trained proprietary ChatGPT-4o and CogVLM2 MLLMs. ChatGPT-4o is used for **P** task execution, while CogVLM2 is used for **R** task execution.
- **Pre-trained LLaVA-OneVision (LLaVA) [36]:** This is a pre-trained proprietary vehicle-specific LLaVA-OneVision MLLM. It consists of a 0.5B-parameter model (LLaVA-0.5B) and a 7B-parameter model (LLaVA-7B).
- **Pre-trained Qwen2-VL (Qwen) [37]:** This is a pre-trained proprietary vehicle-specific Qwen2-VL-Instruct MLLM. It consists of a 2B-parameter model (Qwen-2B) and a 7B-parameter model (Qwen-7B).
- **Fine-tuned GPT-4o+CogVLM2 (GPTCOG-FT):** This is a fine-tuned version of the state-of-the-art vehicle-specific MLLM implemented in [8]. ChatGPT-4o is adapted with few-shot learning (k=2) to **D1** for **P** task execution, while CogVLM2 is fine-tuned with **D2** for **R** task execution.
- **Fine-tuned LLaVA-OneVision (LLaVA-FT):** This is a vehicle-specific LLaVA-OneVision MLLM fine-tuned on **D1** and **D2**. It consists of a 0.5B-parameter model (LLaVA-0.5B-FT) and a 7B-parameter model (LLaVA-7B-FT).
- **Fine-tuned Qwen2-VL (Qwen-FT):** This is a vehicle-specific Qwen2-VLM fine-tuned on **D1** and **D2** (i.e., also referred to as *Ours*). It consists of a 2B-parameter model (Qwen-2B-FT) and a 7B-parameter model (Qwen-7B-FT).

To ensure representative coverage across all datasets, we stratify the evaluation datasets as 20% random splits for each and remove 30% of $\mathbf{F_{HL}}$ from **D1** to mimic real-world scenarios. This ensures that even if there are data collection errors (e.g., missing sensor data), **P** task execution does not over-rely on a single modality or input. Unless explicitly specified, the training and fine-tuning setups and approach apply to all comparative MLLMs.

3) Fine-tuning: For LLaVA-0.5B-FT and our Qwen-2B-FT models, fine-tuning was performed on High-Performance Computing (HPC) hardware equipped with two NVIDIA Tesla V100 Tensor Core GPUs, each with 32GB of VRAM. For LLaVA-7B-FT and our Qwen-7B-FT models, we used an AWS EC2 g5.12xlarge instance with 4 NVIDIA A10G Tensor Core GPUs, each with 24GB of VRAM. Due to the substantial memory requirements of MLLMs, we used Low-Rank Adaptation (LoRA), which introduces controllable low-rank trainable parameters. It enables the training process to fit within available GPU memory. For LLaVA-7B-FT and our Qwen-7B-FT models, we applied 4-bit quantization of trainable parameters using the BitsAndBytes quantization technique. This significantly reduces memory usage while preserving performance. To further optimize the training process, we used the DeepSpeed framework, specifically its Stage 3 offload optimization, to partition model states across GPUs. This helps to minimize memory usage and accelerate training. Optimization was performed using the AdamW optimizer with decoupled weight decay, coupled with a linear learning-rate warm-up and cosine decay to stabilize convergence. Input sequences were dynamically padded and truncated to a fixed maximum length for efficient batching, and gradient accumulation was employed to achieve larger effective batch sizes under memory constraints.

B. Vehicular MLLM Results and Analysis

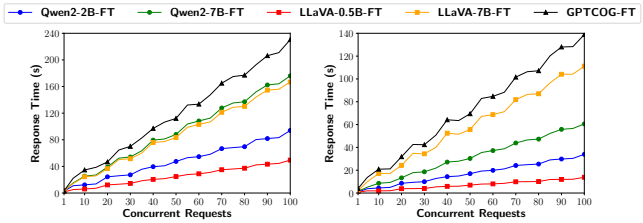
1) Performance Comparison Across Different MLLMs: We evaluate the performance of our Qwen-FT (Qwen-2B-FT/Qwen-7B-FT) against pre-trained GPT-4o-mini, GPTCOG, LLaVA (LLaVA-0.5B/LLaVA-7B), Qwen2-VLM (Qwen-2B/Qwen-7B), and fine-tuned GPTCOG-FT, LLaVA-FT (LLaVA-0.5B-FT/LLaVA-7B-FT) in terms of **P** and **R** task execution. We use BERTScore and Google BLEU (GLEU) as the primary evaluation metrics. BERTScore evaluates text similarity with a range of [0,1] and comprises Precision (p), Recall (r), and

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

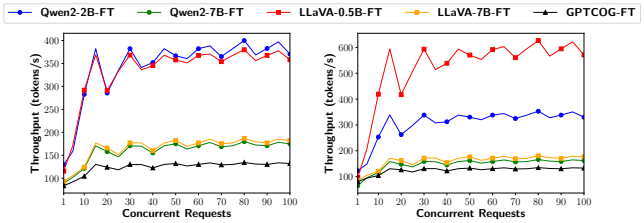
TABLE I: Performance Comparison Across Different MLLMs.

Tasks	Perception (P)				Recommendation (R)			
Metrics	**Precision (p)**	**Recall (r)**	**F1 Score (f1)**	**GLEU (g)**	**Precision (p)**	**Recall (r)**	**F1 Score (f1)**	**GLEU (g)**
Pre-trained MLLMs								
GPT-4o-mini [35]	0.896	0.897	0.897	0.513	0.851	0.877	0.864	0.442
GPTCOG [8]	0.897	0.897	0.897	0.521	0.803	0.817	0.810	0.291
LLaVA-0.5B [36]	0.755	0.767	0.760	0.130	0.711	0.728	0.719	0.084
Qwen-2B [37]	0.871	0.882	0.876	0.432	0.805	0.824	0.814	0.288
LLaVA-7B [36]	0.648	0.669	0.658	0.041	0.796	0.866	0.829	0.281
Qwen-7B [37]	0.896	0.891	0.893	0.520	0.817	0.842	0.829	0.357
Fine-tuned MLLMs								
GPTCOG-FT [8]	0.900	0.904	0.902	0.523	0.889	0.867	0.878	0.400
LLaVA-0.5B-FT	0.835	0.824	0.829	0.269	0.852	0.851	0.851	0.259
Qwen-2B-FT (Ours)	**0.928**	**0.926**	**0.927**	**0.584**	**0.895**	**0.876**	**0.885**	**0.458**
LLaVA-7B-FT	0.841	0.833	0.836	0.273	0.883	0.857	0.869	0.313
Qwen-7B-FT (Ours)	**0.935**	**0.926**	**0.931**	**0.608**	**0.899**	**0.878**	**0.888**	**0.460**



(a) Perception task, **P** (b) Recommendation task, **R**

Fig. 3: Concurrent requests vs. response time.



(a) Perception task, **P** (b) Recommendation task, **R**

Fig. 4: Concurrent requests vs. throughput.

F1-score (f1) [38]. GLEU measures the overlap of n-grams (sequence of words) between the generated and reference texts and comprises the GLEU score (g) [39]. Columns 2-5 and Columns 6-9 of Table I provide the performance of the various MLLMs in terms of **P** and **R** task execution, respectively.

From Table I, we observe that the 7B parameter models generally outperform their 0.5B or 2B counterparts in both tasks, illustrating the benefits of increased capacity for learning complex patterns. However, an exception arises with LLaVA-0.5B, which surpasses its 7B counterpart in **P** task execution. This anomaly could stem from sampling-based token generation, where the 7B model's broader token probability distribution leads to greater variability in token selection than the 0.5B model. From a fine-tuning viewpoint, both LLaVA-0.5B-FT and our Qwen-2B-FT outperform their pre-trained versions. GPTCOG-FT achieves better results than GPTCOG for **P** and **R** tasks. Overall, our Qwen-FT outperforms GPTCOG-FT and LLaVA-FT models across both parameter scales and tasks. For instance, our Qwen-7B-FT achieves p, r, f1, and g scores of 0.935, 0.926, 0.931, and 0.608, respectively, in **P** task compared to lower scores for GPTCOG-FT and LLaVA-7B-FT. This highlights its robust architecture for handling multimodal inputs. In conclusion, the fine-tuned models outperform their pre-trained counterparts across the board. This confirms that specialized training enhances performance on multimodal tasks. The superiority of our Qwen in both **P** and **R** tasks affirms its suitability for the application scenario in this work.

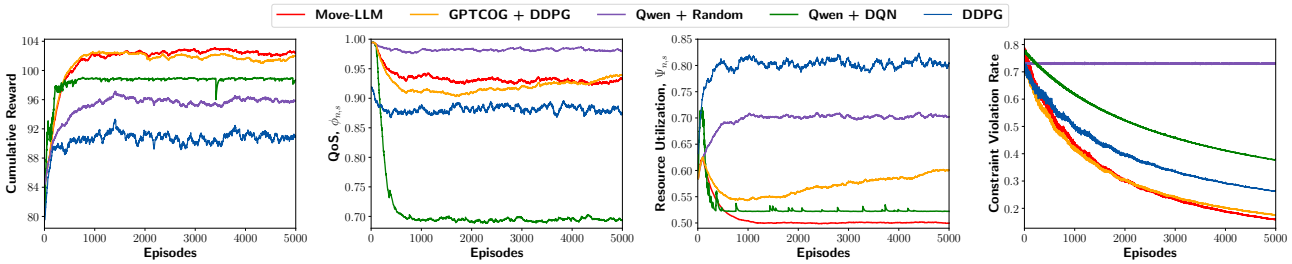
2) Performance on Concurrent Requests: In this simulation, we evaluate the performance of our Qwen-2B-FT and Qwen-7B-FT against LLaVA-0.5B-FT, LLaVA-7B-FT, and GPTCOG-FT in terms of response time (in s) and throughput (in tokens/s) under a varying number of concurrent requests. We submit 1-100 video segments requests and their accompanying metadata for inference (**P** tasks). A similar batch of requests is also submitted to test the models' recommendation (**R** tasks) capabilities. Each MLLM is deployed with 4 model replicas, each hosted on a GPU with 32GB VRAM. Fig. 3 and Fig. 4 illustrate the results for response time and throughput of **P** and **R** tasks, respectively.

From Fig. 3, we observe that the response time increases with an increasing number of concurrent requests across all MLLMs for both **P** and **R** tasks. The increase is more pronounced in GPTCOG-FT and the 7B models, which require more computational resources due to their larger parameter sizes. For the smaller models, we observe that LLaVA-0.5B-FT achieves lower response times than our Qwen-2B-FT across both tasks. This is because LLaVA-0.5B-FT has fewer parameters and thus requires fewer computations. Among the 7B models, LLaVA-7B-FT outperforms our Qwen-7B-FT slightly for **P** task in Fig. 3(a), while our Qwen-7B-FT reports a pronounced outperformance over LLaVA-7B-FT in **R** task in Fig. 3(b) in terms of lower response time. This advantage may be attributed to the model's structure and optimization strategies, which allow LLaVA-7B-FT to handle concurrent requests more effectively, despite having the same size as our Qwen-7B-FT, in **P** task execution. However, our Qwen-7B-FT is more suitable for **R** task execution. It can be observed that the average response time for a complete **P-R** task execution is in a few seconds. A well-designed Retrieval Augmented Generation (RAG) mechanism could reduce response time to the millisecond range, a requirement for 6G vehicular networks.

As shown in Fig. 4, throughput initially increases with increasing concurrent requests but fluctuates and becomes steady after a few request counts. From Fig. 4(a), LLaVA-7B-FT achieves higher throughput than GPTCOG-FT and our Qwen-7-FT for **P** tasks, which is the converse for their smaller versions. This is because our Qwen may have superior architectural features, allowing it to scale better even under high concurrency. This is evident in our Qwen-2B-FT, which achieves similar results as a

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987



(a) Cumulative Reward (b) QoS (c) Resource Utilization (d) Constraint Violation Rate

Fig. 5: Convergence analysis.

much bigger model compared to LLaVA-0.5B-FT. Additionally, **P** tasks involve multimodal data fusion, where our Qwen's optimized design can handle large amounts of input data more effectively. From Fig. 4(b), LLaVA-0.5B-FT achieves a much higher throughput than our Qwen-2B-FT for **R** tasks, while the trend for the larger models reflects that of the **P** tasks. **R** tasks involve simpler input-output relationships, where LLaVA's model structure is better suited than our Qwen. Moreover, our Qwen's underutilized vision encoder significantly reduces its throughput. In summary, we conclude that our Qwen-FT can achieve moderate and acceptable response time and throughput while achieving higher accuracy in **P** and **R** tasks compared with the comparative LLaVA-FT and GPTCOG-FT.

C. MLLM-assisted DDPG (Move-LLM) Performance Evaluation

In this subsection, we evaluate the performance of our proposed MLLM-assisted DDPG (**Move-LLM**) via extensive simulations. **Move-LLM** comprises Qwen-7B-FT (our best model) and an improved DDPG algorithm for end-to-end **P-R-D** task execution. All simulations are implemented using Python 3.12 with key libraries such as PyTorch 2.5.1 and NumPy installed. The simulations are executed on a computer that accesses HPC hardware of two NVIDIA Tesla V100 Tensor Core GPUs, each providing 32GB of VRAM. The underlying DDPG framework uses two fully connected FFN layers for both the actor and critic networks (128 neurons per hidden layer). Target networks are maintained for both the actor and the critic to stabilize training, and are softly updated using Polyak averaging. We set γ, $\boldsymbol{M}$, and $\boldsymbol{M}'$ to 0.99, 5000, and 16, respectively. We also set the learning rate to 10^{-3}, weight decay to 10^{-2}, and μ to 0.001 for both actor and critic networks. We use the *ReLU* activation function for the hidden layers, *sigmoid* for action bounds, and *AdamOptimizer* to optimize the loss.

Comaprative Methods: compare the performance of **Move-LLM** with the following benchmarks:

a) GPTCOG + DDPG: GPTCOG + DDPG takes the output of the GPTCOG-FT's **R** task as a partial input to its DDPG's state space.

b) Qwen + DQN: DQN is a value-based algorithm that approximates the Q-value function and is well-suited for discrete action spaces [40]. It struggles with high-dimensional continuous control problems because it relies on action-value discretization. Qwen + DQN takes the output of Qwen-7B-FT's **R** task as a partial input to its DQN state space.

c) Qwen + Random: This algorithm selects actions uniformly from the action space without learning from past experiences and serves as a lower-bound benchmark for the MLLM-assisted DRL algorithms. Qwen + Random takes the output of Qwen-7B-FT's **R** task as a partial input.

d) Vanilla DDPG (DDPG): DDPG is an actor-critic method that concurrently learns a Q- function and a policy function [30]. Unlike other comparative methods, this method does not use Qwen-7B-FT's **R** task results as its input state space. Instead, it aims to evaluate the contextual leverage gained by Qwen-7B-FT. The observation space comprises **V** and $\mathbf{F_{HL}}$, which are transformed into either categorical or continuous representations. The actor network input is processed through a feature encoder that integrates a ResNet3D model [41] for visual input (**V**) and an MLP for the corresponding metadata. Therefore, this DDPG method serves as a non-MLLM-assisted DRL baseline.

D. Move-LLM Results and Analysis

1) Convergence Analysis: In this simulation, we evaluate the convergence performance of the proposed **Move-LLM** in comparison to GPTCOG + DDPG, Qwen + DQN, Qwen + Random, and DDPG algorithms in terms of the cumulative reward. We run the simulation for 5000 episodes, and a reward is computed at each decision step after an action is sampled from the policy. Since cumulative reward fluctuates significantly per episode, we apply an exponential moving average to smooth the plots. Fig. 5 presents the convergence trends of the five comparative algorithms. The weighting factors are set to $\alpha_1 = 0.75$ and $\alpha_2 = 0.25$ to balance QoS and RU.

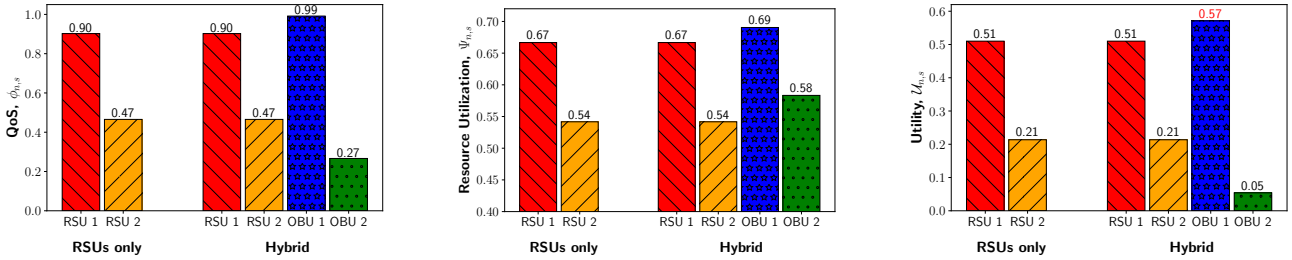
From Fig. 5(a), we observe that all algorithms converge around 1200 episodes, with their cumulative rewards stabilizing at approximately 103, 102, 98, 95, and 91 for **Move-LLM**, GPTCOG + DDPG, Qwen + DQN, Qwen + Random, and DDPG, respectively. **Move-LLM** achieves the highest cumulative reward, indicating its superior ability to learn an effective continuous control policy. GPTCOG + DDPG initially performs better than **Move-LLM**, but worsens slightly after convergence, as it optimizes a stochastic policy with entropy regularization, sacrificing immediate performance for long-term robustness. Qwen + DQN exhibits lower cumulative reward among the MLLM-assisted learning-based algorithms due to its reliance on discrete action selection; discretizing the action spaces disrupts pertinent information. Qwen + Random policy performs worse than the MLLM-assisted learning-based methods because it lacks a learning policy. Notably, DDPG (non-MLLM-assisted DDPG) performs the worst overall. This decline is attributed to the high-dimensional state space, which lacks the structured assistance provided by the MLLM. Without MLLM-assisted feature extraction or representation, DDPG struggles with generalization and is more sensitive

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted,
but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

(a) Expected Utility (b) QoS (c) Resource Utilization

Fig. 6: Error propagation analysis.



(a) QoS (b) Resource Utilization (c) Expected utility

Fig. 7: Ablation study.

to the design of the input space, requiring careful manual crafting to handle the raw environment observations effectively.

In Fig. 5(b) and 5(c), we normalized $\phi_{n,s}$ and $\Psi_{n,s}$ while comparing the convergence trends of the five algorithms. From Fig. 5(b), we observe that Qwen + Random achieves the highest QoS at about 0.99 as compared to **Move-LLM**, GPTCOG + DDPG, Qwen + DQN, and DDPG at approximately 0.94, 0.90, 0.70, and 0.87, respectively. Although the $\phi_{n,s}$ values remain above 0.85 for most algorithms, Qwen + DQN shows a notable drop. However, analyzing the $\Psi_{n,s}$ trends in Fig. 5(c) reveals that DDPG achieves the highest $\Psi_{n,s}$ ($\sim 80\%$), whereas **Move-LLM** achieves the lowest ($\sim 50\%$) translating into the fact that DDPG consumes more resources to achieve a competitive QoS. Given that $\mathcal{U}_{n,s}$ seeks to achieve a trade-off between high $\phi_{n,s}$ and low $\Psi_{n,s}$, we can conclude that **Move-LLM** proves to be the most effective algorithm that optimizes $\phi_{n,s}$ and $\Psi_{n,s}$ while achieving the highest cumulative reward.

To report constraint violation rates of C2 and C4, we extend the simulations on convergence by counting violations per episode until the service is deployed or the maximum number of steps is reached. All simulation settings follow that of Fig. 5(a)-(c). For ease of presentation, we add the constraint violation counts for C2 and C4 at each episode and normalize them to the range [0, 1]. Fig. 5(d) shows the normalized cumulative violation rate per episode. The results indicate that **Move-LLM** consistently achieves the lowest long-term violation rate among all comparative methods. This demonstrates better constraint satisfaction over a longer period of operation.

Overall, we observe that the performance gains of **Move-LLM** stem from integrating Qwen-7B-FT (for perception and recommendation) with an improved DDPG (for deployment) into a unified MLLM-assisted DDPG framework for on-demand vehicular service deployment. This approach outperforms other MLLM-assisted DRL baselines, the MLLM-assisted heuristic baseline, and the non-MLLM-assisted DRL baseline. Therefore, we conclude that the performance gains come from both the rich contextual model inference and service recommendation outputs from the vehicular MLLM, as well as the superiority of the improved DDPG algorithm.

2) Error Propagation Analysis: In this simulation, we perform an error propagation analysis to verify that Rec_s from the vehicular MLLM does not lead to extensive DRL policy failure. We evaluate the performance of **Move-LLM** on resource allocation with respect to expected utility $\mathcal{U}_{n,s}$, QoS $\phi_{n,s}$, and resource utilization $\Psi_{n,s}$, comparing it with the ground-truth. We consider three distinct service types for each recommendation output: Service 1, Service 2, and Service 3. Fig. 6 reports the error propagation analysis results of **Move-LLM** and the ground truth. From the figure, we observe that there is no significant difference between the performance of **Move-LLM** and the ground truth. This is because services from Rec_s and the ground truth are within the same category and have similar QoS and resource utilization requirements. Thus, their corresponding resource allocations and expected utilities are also similar. We can conclude that the output of the vehicular MLLM does not cause significant DRL policy failure.

3) Performance on Node Selection and Resource Allocation: Based on the output of the **R** task, we select an optimal node amongst RSUs and OBU clusters and allocate optimal resources for the **D** task. For node selection, 2 RSUs and 2 OBU clusters are randomly positioned relative to the location of the traffic event for service deployment. For ease of presentation in results, we eval-

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

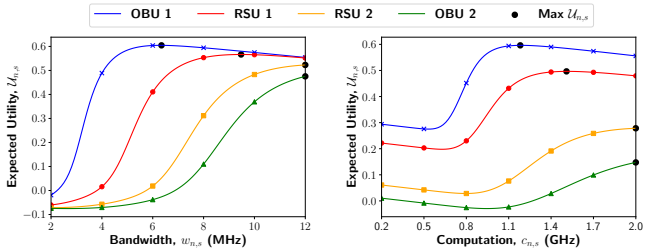
This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

uate the performance of **Move-LLM** by deploying an *Emergency Medical Service* within Rec_s. The service is characterized by the following req_s properties and requirements: $r_s^{min} = 50$Mbps, $\varpi_s^{max} = 40$ms, $l_s = 128$kB, and $\varsigma_s = 24$ Megacycles. All other parameters are chosen according to [5].

a) Ablation study: To evaluate the impact of integrating OBU clusters as alternative hosting nodes into the service deployment framework, we conduct an ablation study under two deployment scenarios: (i) RSU-only deployment and (ii) hybrid (RSU + OBU clusters) deployment. To ensure a fair comparison, the service requirements of the selected service remain unchanged across all node options. We set $w_{n,s}$ and $c_{n,s}$ to 8.0 MHz and 1.0 GHz, respectively, for all node options to achieve the desirable results. We select different node configurations for each node such that RSU 2 and OBU Cluster 2 (OBU 2) require comparatively more resources to achieve a reasonable QoS. Additionally, OBU 2 has limited resource availability, making it a more constrained candidate for service deployment. Fig. 7 shows the results of the ablation study in terms of expected QoS, RU, and utility.

From Fig. 7(a), we observe that under the RSU-only deployment configuration, the maximum achievable QoS is 0.90. However, incorporating OBU clusters increases the maximum achievable QoS to 0.99, a 10% increase over the RSU-only deployment. As shown in Fig. 7(b), RSU 2 and OBU 2 exhibit the lowest RU at about 0.54 and 0.58, respectively. This explains why they achieve lower QoS since they require more resources to achieve reasonable QoS values. OBU 1 and RSU 1 achieve higher RU at approximately 0.67 and 0.69, respectively, at the expense of higher QoS compared with RSU 2 and OBU 2. Notably, OBU 1 records the worst RU (69%), which is an acceptable RU level considering that it achieves the highest QoS. The hybrid deployment option consumes at least 3% more resources compared with the RSU-only deployment. This percentage increase could be considered acceptable, as it yields at least a 10% improvement in QoS. Fig. 7(c) presents the results on the expected utility $\mathcal{U}_{n,s}$, where OBU 1 achieves the highest utility value of 0.57 compared with 0.51, 0.21, and 0.05 by RSU 1, RSU 2, and OBU 2, respectively. OBU 2 achieves the worst $\mathcal{U}_{n,s}$ due to its limited resource availability. Thus, the hybrid deployment mode achieves at least 11.76% increase in $\mathcal{U}_{n,s}$ as compared to the RSU-only deployment option. This trend in results suggests a favorable tradeoff between QoS and RU, validating the effectiveness of the hybrid node deployment option. This demonstrates the benefit of having OBU clusters as alternative deployment node options, offering dynamic node mobility and flexibility.

b) Node selection: Fig. 8 shows the performance of **Move-LLM** in terms of normalized expected utility $\mathcal{U}_{n,s}$ with increasing $w_{n,s}$ and $c_{n,s}$. We use the same simulation configuration as in the ablation study. From Fig. 8(a), we keep $c_{n,s}$ constant at $c_{n,s} = 1.13GHz$ and vary $w_{n,s}$ as $w_{n,s} = [2, 4, 6, 8, 10, 12]$ MHz. From the figure, we observe that $\mathcal{U}_{n,s}$ for each node increases with $w_{n,s}$ until it reaches a maximum, then declines. Notably, OBU 1 achieves the highest $\mathcal{U}_{n,s}$ with approximately 6 MHz bandwidth, while RSU 1, RSU 2, and OBU 2 achieve their highest $\mathcal{U}_{n,s}$ with about 9 MHz, 12 MHz, and 12 MHz, respectively. This implies that OBU 1 utilizes its bandwidth resource more efficiently to balance QoS and RU while keeping bandwidth allocation at acceptable levels. Although the other nodes use more bandwidth, their $\mathcal{U}_{n,s}$ are lower because they cannot achieve the best trade-off between QoS and RU. A similar trend is achieved in Fig. 8(b), where we keep $w_{n,s}$ constant at $w_{n,s} = 6.4MHz$ and vary $c_{n,s}$ as $c_{n,s} = [0.2, 0.5, 0.8, 1.1, 1.4, 1.7, 2.0]$ GHz. The results suggest that once Algorithm 1 forms OBU clusters and makes them available for on-demand service hosting, they can be competitive, and even superior alternatives to RSUs, particularly for delay-sensitive services. This highlights their potential in supporting a dynamic and efficient vehicular environment.



(a) $w_{n,s}$ vs. $\mathcal{U}_{n,s}$ (b) $c_{n,s}$ vs. $\mathcal{U}_{n,s}$

Fig. 8: Performance on node selection.

c) Resource allocation: To isolate the effect of resource allocation from node selection, we select OBU Cluster 1 for resource allocation in this simulation, with the following $resp_n$ properties: $\boldsymbol{W}_n = 12$ MHz, $\boldsymbol{C}_n = 2.0$ GHz, and $SNR_n = 48.2$. The minimum $w_{n,s}$ and $c_{n,s}$ are set to 2 MHz and 0.2 GHz, respectively. We evaluate the performance of the five comparative methods in terms of $\mathcal{U}_{n,s}$, $\phi_{n,s}$ and $\Psi_{n,s}$ with their corresponding $w_{n,s}$ and $c_{n,s}$, as shown in Fig. 9. The **x** marker in all plots represents the ideal point without resource constraints. The maximum $\phi_{n,s}$, normalized to 1.0, is achieved when all available resources are fully allocated, while the ideal $\Psi_{n,s}$ corresponds to minimum resource usage.

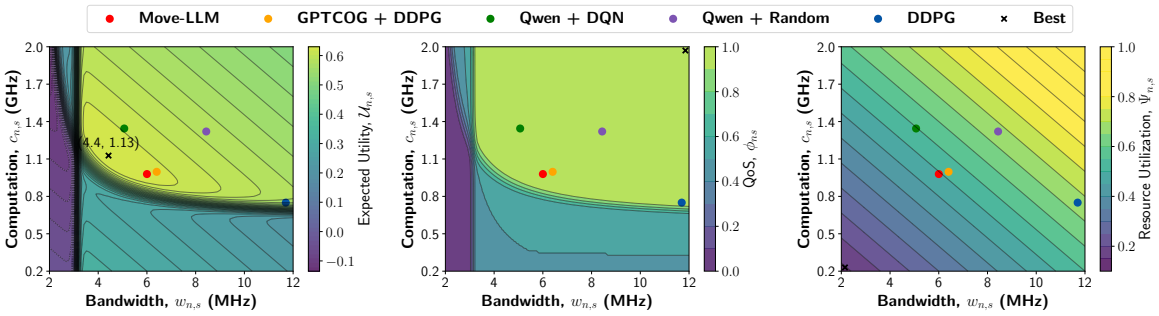
From Fig. 9(a), we observe that the ideal $\mathcal{U}_{n,s}$ occurs at $(w^*_{n,s}, c^*_{n,s}) = (4.4, 1.13)$ with $\mathcal{U}^*_{n,s} = 0.63$. **Move-LLM** achieves resource allocation close to the optimal with $(w_{n,s}, c_{n,s}) = (6.0, 1.0)$ and $\mathcal{U}_{n,s} = 0.61$ while GPTCOG + DDPG, Qwen + DQN, Qwen + Random, and DDPG follow with about $(w_{n,s}, c_{n,s}) = (6.5, 1.1)$, $(w_{n,s}, c_{n,s}) = (3.0, 1.4)$, $(w_{n,s}, c_{n,s}) = (9.0, 1.3)$, and $(w_{n,s}, c_{n,s}) = (11.7, 0.8)$, and $\mathcal{U}_{n,s} = 0.60$, $\mathcal{U}_{n,s} = 0.59$, $\mathcal{U}_{n,s} = 0.57$, and $\mathcal{U}_{n,s} = 0.55$ respectively. **Move-LLM**'s allocation closely aligns with the optimal values, lying in the contour region of highest utility, while GPTCOG + DDPG and Qwen + DQN remain around the vicinity of that contour. This aligns with the trends in Fig. 5, where despite variations due to exploration noise and the presence of multiple services during training, the overall trade-off pattern remains consistent.

From Fig. 9(b), we observe that **Move-LLM** achieves similar results as the comparative methods, as they all appear to be in the same contour. Notably, all algorithms allocate different $w_{n,s}$ and $c_{n,s}$ but achieve similar $\phi_{n,s}$. From Fig. 9(c), we observe that **Move-LLM** achieves the least $\Psi_{n,s}$ as compared to the comparative methods, indicating its superiority in achieving the best $\mathcal{U}_{n,s}$ with the optimal $\rho_{n,s}$. We can conclude that **Move-LLM** surpasses the other algorithms in maximizing $\phi_{n,s}$ while minimizing $\Psi_{n,s}$, to achieve the highest $\mathcal{U}_{n,s}$.

d) Impact of time-varying network conditions: To address the impact of time-varying network conditions, we extend the analysis in Fig. 9 by explicitly modeling vehicle mobility dynamics

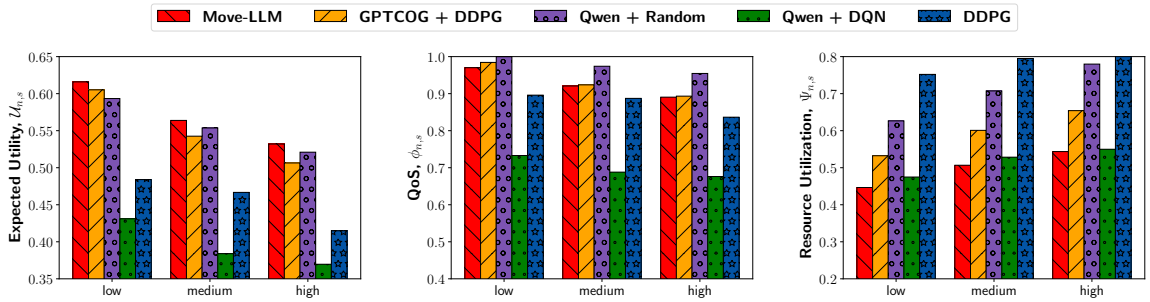
Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987



(a) Utility Reward (b) QoS (c) Resource Utilization

Fig. 9: Performance on resource allocation.



(a) Expected Utility (b) QoS (c) Resource Utilization

Fig. 10: Impact of time-varying network conditions.

through a time-of-day–based evaluation. User requests within a daily interval are divided into hourly segments and grouped into three mobility categories that naturally capture channel variability as low, medium, and high mobility. Low mobility corresponds to rush hours (7–9 AM and 5–7 PM), characterized by high congestion and relatively static vehicle movement, with an average density of 25 vehicles per 100 m². Medium mobility represents off-peak daytime periods (10-11 AM and 2-4 PM) with moderate traffic flow, averaging 15 vehicles per 100 m², while high mobility corresponds to late-night (9-11 PM) or early-morning (4-6 AM) periods with sparse traffic and fast-moving vehicles, averaging 7 vehicles per 100 m². These categories implicitly reflect time-varying channel conditions, as higher mobility leads to more rapid channel fluctuations. As shown in Fig. 10, lower mobility leads to higher utility and QoS because more stable OBU clustering reduces handoff interruptions and channel variability, enabling DRL policies to converge more effectively under slowly varying channel states. RU increases with mobility since frequent topology changes and handoffs require allocating additional resources to maintain service continuity. Across most mobility regimes, **Move-LLM** consistently outperforms other comparative methods. This demonstrates robust convergence and strong generalizability under diverse, time-varying, and mobility-driven network conditions.

VI. Conclusion

This paper proposed **Move-LLM**, a novel MLLM-assisted DDPG framework for context-aware on-demand service deployment in 6G vehicular networks. Our framework comprises a vehicular MLLM that performs **P** and **R** tasks, and a DRL agent that performs **D** tasks. Notably, the vehicular MLLM is fine-tuned on multimodal vehicular event datasets to provide a more comprehensive, context-aware understanding of vehicular network traffic, thereby enhancing the accuracy of on-demand service recommendations for specific traffic events. We introduced OBU clusters as alternatives to fixed RSUs for dynamic service hosting. Then, we developed an improved DDPG-based node selection and resource allocation algorithm that optimizes on-demand service deployment considering resource allocation, expected utility, and resource limitations of each RSU and OBU cluster. Simulation results demonstrated the superiority of the proposed vehicular MLLM compared with GPT-4o-mini. Furthermore, **Move-LLM** consistently identified an optimal deployment node and allocated optimal resources, ensuring efficient and reliable service delivery. Future work will explore cloud-edge collaboration of MLLM deployment for service hosting. Additionally, future work will explore the feasibility of RAG-based mechanisms to reduce the response time of fine-tuning tasks and meet 6G ultra-low latency requirements.

References

[1] R. Liu, M. Hua, K. Guan, X. Wang, L. Zhang, T. Mao, D. Zhang, Q. Wu, and A. Jamalipour, "6g enabled advanced transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 9, pp. 10 564–10 580, 2024.

[2] H. Sami, A. Mourad, and W. El-Hajj, "Vehicular-obus-as-on-demand-fogs: Resource and context aware deployment of containerized micro-services," *IEEE/ACM trans. on networking*, vol. 28, no. 2, pp. 778–790, 2020.

[3] W. Mao, O. U. Akgul, B. Cho, Y. Xiao, and A. Ylä-Jääski, "On-demand vehicular fog computing for beyond 5g networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 12, pp. 15 237–15 253, 2023.

[4] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-defined networking for rsu clouds in support of the internet of vehicles," *IEEE Internet of Things journal*, vol. 2, no. 2, pp. 133–144, 2014.

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted,
but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

[5] Y. Huang, B. Feng, Y. Cao, Z. Guo, M. Zhang, and B. Zheng, "Collaborative on-demand dynamic deployment via deep reinforcement learning for iov service in multi edge clouds," *Journal of Cloud Comp.*, vol. 12, no. 1, p. 119, 2023.

[6] H. Sami, R. Saado, A. El Saoudi, A. Mourad, H. Otrok, and J. Bentahar, "Opportunistic uav deployment for intelligent on-demand iov service management," *IEEE Trans. on Net. and Serv. Mgt.*, vol. 20, no. 3, pp. 3428–3442, 2023.

[7] Q. Zhang, M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Machine learning for predictive on-demand deployment of uavs for wireless communications," in *2018 IEEE Global Commun. Conf. (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[8] Y. Hu, D. Ye, J. Kang, M. Wu, and R. Yu, "A cloud–edge collaborative architecture for multimodal llm-based advanced driver assistance systems in iot networks," *IEEE Internet of Things Journal*, vol. 12, no. 10, pp. 13 208–13 221, 2025.

[9] B. Rong and H. Rutagemwa, "Leveraging large language models for intelligent control of 6g integrated tn-ntn with iot service," *IEEE Network*, vol. 38, no. 4, pp. 136–142, 2024.

[10] M. Fu, P. Wang, M. Liu, Z. Zhang, and X. Zhou, "Iov-bert-ids: Hybrid network intrusion detection system in iov using large language models," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 2, pp. 1909–1921, 2025.

[11] C. Cui, Y. Ma, X. Cao, W. Ye, Y. Zhou, K. Liang, J. Chen, J. Lu, Z. Yang, K.-D. Liao, T. Gao, E. Li, K. Tang, Z. Cao, T. Zhou, A. Liu, X. Yan, S. Mei, J. Cao, Z. Wang, and C. Zheng, "A survey on multimodal large language models for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, January 2024, pp. 958–979.

[12] M. Ammous, S. Belakaria, S. Sorour, and A. Abdel-Rahim, "Optimal cloud-based routing with in-route charging of mobility-on-demand electric vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 7, pp. 2510–2522, 2019.

[13] H. Sami, R. Saado, A. E. Saoudi, A. Mourad, H. Otrok, and J. Bentahar, "Opportunistic uav deployment for intelligent on-demand iov service management," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3428–3442, 2023.

[14] L. Wang, H. Zhang, S. Guo, D. Li, and D. Yuan, "Learning to deployment: Data-driven on-demand uav placement for throughput maximization," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 6, pp. 8007–8012, 2024.

[15] G. O. Boateng, H. Sami, A. Alagha, H. Elmekki, A. Hammoud, R. Mizouni, A. Mourad, H. Otrok, J. Bentahar, S. Muhaidat, C. Talhi, Z. Dziong, and M. Guizani, "A survey on large language models for communication, network, and service management: Application insights, challenges, and future directions," *IEEE Communications Surveys Tutorials*, vol. 28, pp. 527–566, 2026.

[16] S. Fang, J. Liu, M. Ding, Y. Cui, C. Lv, P. Hang, and J. Sun, "Towards interactive and learnable cooperative driving automation: a large language model-driven decision-making framework," *IEEE Transactions on Vehicular Technology*, pp. 1–12, 2025.

[17] R. Chen, W. Song, W. Zu, Z. Dong, Z. Guo, F. Sun, Z. Tian, and J. Wang, "An llm-driven framework for multiple-vehicle dispatching and navigation in smart city landscapes," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 2147–2153.

[18] C. Liu and J. Zhao, "Resource allocation in large language model integrated 6g vehicular networks," in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, 2024, pp. 1–6.

[19] H. Liu, R. Yao, Z. Huang, S. Shen, and J. Ma, "Lmmcodrive: Cooperative driving with large multimodal models," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 10 532–10 539.

[20] Y. He, J. Fang, F. R. Yu, and V. C. Leung, "Large language models (llms) inference offloading and resource allocation in cloud-edge computing: An active inference approach," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 253–11 264, 2024.

[21] Y. Ding, C. Niu, F. Wu, S. Tang, C. Lyu, and G. Chen, "Enhancing on-device llm inference with historical cloud-based llm interactions," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 597–608. [Online]. Available: https://doi.org/10.1145/3637528.3671679

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[23] S. R. A. Sebakara, G. Sun, G. O. Boateng, B. Mareri, R. Ou, and G. Liu, "Snaf: Drl-based interdependent e2e resource slicing scheme for a virtualized network," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 9069–9084, 2023.

[24] Z. Wang, Y. Wei, F. R. Yu, and Z. Han, "Utility optimization for resource allocation in edge network slicing using drl," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[25] C. Xu, T. Li, M. Sheng, and J. Li, "Self-organized dynamic caching space sharing in virtualized wireless networks," in *2016 IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1–6.

[26] L. Liberti, "Undecidability and hardness in mixed-integer nonlinear programming," *RAIRO Oper. Res.*, vol. 53, pp. 81–109, 2019.

[27] J. Luo, J. Song, F.-C. Zheng, L. Gao, and T. Wang, "User-centric uav deployment and content placement in cache-enabled multi-uav networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5656–5660, 2022.

[28] H. Sami and A. Mourad, "Dynamic on-demand fog formation offering on-the-fly iot service deployment," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1026–1039, 2020.

[29] G. O. Boateng, H. Si, H. Xia, X. Guo, C. Chen, I. O. Agyemang, and N. Ansari, "Automated valet parking and charging: A dynamic pricing and reservation-based framework leveraging multi-agent reinforcement learning," *IEEE Transactions on Intelligent Vehicles*, vol. 10, no. 2, pp. 1010–1029, 2025.

[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019. [Online]. Available: https://arxiv.org/abs/1509.02971

[31] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6382–6393.

[32] A. Y. et al., "Qwen2 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2407.10671

[33] L. Kezebou, V. Oludare, K. Panetta, J. Intriligator, and S. Agaian, "Highway accident detection and classification from live traffic surveillance cameras: a comprehensive dataset and video action recognition benchmarking," in *Multimodal Image Exploitation and Learning 2022*, S. S. Agaian, V. K. Asari, S. P. DelMarco, and S. A. Jassim, Eds., vol. 12100, May 2022, p. 121000M.

[34] A. Chan and N. Vasconcelos, "Probabilistic kernels for the classification of auto-regressive visual processes," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 846–851 vol. 1.

[35] OpenAI, "Gpt-4o mini: advancing cost-efficient intelligence," 2024. [Online]. Available: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[36] B. Li, Y. Zhang, D. Guo, R. Zhang, F. Li, H. Zhang, K. Zhang, P. Zhang, Y. Li, Z. Liu, and C. Li, "Llava-onevision: Easy visual task transfer," 2024. [Online]. Available: https://arxiv.org/abs/2408.03326

[37] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, Y. Fan, K. Dang, M. Du, X. Ren, R. Men, D. Liu, C. Zhou, J. Zhou, and J. Lin, "Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution," 2024. [Online]. Available: https://arxiv.org/abs/2409.12191

[38] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020. [Online]. Available: https://arxiv.org/abs/1904.09675

[39] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016.

[40] M. Roderick, J. MacGlashan, and S. Tellex, "Implementing the deep q-network," 2017. [Online]. Available: https://arxiv.org/abs/1711.07478

[41] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," *CoRR*, vol. abs/1711.11248, 2017. [Online]. Available: http://arxiv.org/abs/1711.11248

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted,
but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2026.3682987

Gordon Owusu Boateng received his Bachelor's degree in Telecommunications Engineering from the Kwame Nkrumah University of Science and Technology (KNUST), Kumasi-Ghana, in 2014. He received his M.Eng. degree and Ph.D. degree in Computer Science and Technology from the University of Electronic Science and Technology of China (UESTC), in 2019 and 2023, respectively. He was a Postdoctoral Researcher with the Hybrid Positioning Research Group (HPRG) at UESTC, in 2023. He is currently a Postdoctoral Fellow at the KU 6G Research Center, Khalifa University, Abu Dhabi, UAE. Gordon has co-authored over 35 scientific journal and conference papers. His research interests include 5G/6G wireless networks, blockchain, reinforcement learning, vehicular networks, and large language models.

Amine Kidane Ghebreziabiher received his Computer Engineering degree from Khalifa University of Science and Technology, Abu Dhabi, UAE, in 2024. He completed his research internship at New York University Abu Dhabi, in 2023. Currently, he is a Research Assistant at the KU 6G Research Center, Khalifa University. His research interests include large language models, 5G/6G wireless networks, vehicular networks, computer vision, and reinforcement learning.

Rabeb Mizouni is an Associate Professor in the Department of Electrical Engineering and Computer Science at Khalifa University, Abu Dhabi, United Arab Emirates. She got her M.Sc. and Ph.D. in Electrical and Computer Engineering from Concordia University, Montreal, Canada in 2002 and 2007 respectively. Currently, she is interested in the deployment of context aware mobile applications, crowd sensing, Artificial Intelligence, IoT and Blockchain. Dr. Mizouni is currently an Associate Editor for IEEE Internet of Things magazine.

Azzam Mourad received his M.Sc. in CS from Laval University, Canada (2003) and Ph.D. in ECE from Concordia University, Canada (2008). He is currently a Professor of Computer Science and Founding Director of the Cyber Security Systems and Applied AI Research Center with the Lebanese American University, Visiting Professor of Computer Science with New York University Abu Dhabi and Affiliate Professor with the Software Engineering and IT Department, Ecole de Technologie Superieure (ETS), Montreal, Canada. His research interests include Cyber Security, Federated Machine Learning, Network and Service Optimization and Management targeting IoT and IoV, Cloud/Fog/Edge Computing, and Vehicular and Mobile Networks. He has served/serves as an associate editor for IEEE Transactions on Services Computing, IEEE Transactions on Network and Service Management, IEEE Network, IEEE Open Journal of the Communications Society, IET Quantum Communication, and IEEE Communications Letters, the General Chair of IWCMC2020-2022, the General Co-Chair of WiMob2016, and the Track Chair, a TPC member, and a reviewer for several prestigious journals and conferences. He is an IEEE senior member.

Hadi Otrok received his Ph.D. in ECE from Concordia University. He holds an Associate Professor position in the department of Electrical Engineering and Computer Science (EECS) at Khalifa University. Also, he is an Affiliate Associate Professor in the Concordia Institute for Information Systems Engineering at Concordia University, Montreal, Canada, and an Affiliate Associate Professor in the Electrical department at Ecole de Technologie Superieure (ETS), Montreal, Canada. His research interests include the domain of blockchain, reinforcement learning, crowd sensing and sourcing, ad hoc networks, and cloud security. He co-chaired several committees at various IEEE conferences. He is also an Associate Editor at IEEE Transactions on Network and Service Management (TNSM), Ad-hoc networks (Elsevier), and IEEE Network. He also served from 2015 to 2019 as an Associate Editor at IEEE Communications Letters.

Jamal Bentahar received the Ph.D. degree in computer science and software engineering from Laval University, Canada, in 2005. He is a Professor with Concordia Institute for Information Systems Engineering, Concordia University, Canada. From 2005 to 2006, he was a Postdoctoral Fellow with Laval University, and then NSERC Postdoctoral Fellow at Simon Fraser University, Canada. He was an NSERC Co-Chair for Discovery Grant for Computer Science (2016-2018). He is a visiting professor at Khalifa University of Science and Technology. His research interests include the areas of computational logics, reinforcement learning, multi-agent systems, service computing, game theory, and software engineering.

Sami Muhaidat received his Ph.D. in Electrical and Computer Engineering from the University of Waterloo, Ontario, in 2006. From 2007 to 2008, he was an NSERC Postdoctoral Fellow in the Department of Electrical and Computer Engineering at the University of Toronto, Canada. From 2008 to 2012, he served as an Assistant Professor in the School of Engineering Science at Simon Fraser University, British Columbia, Canada. Currently, he is a Professor and the Associate Dean for Research in the College of Computing and Mathematical Sciences at Khalifa University. He is also an Adjunct Professor at Carleton University, Ontario, Canada. Sami's research interests include advanced digital signal processing techniques for wireless communications, intelligent surfaces, machine learning for communications, optical communications, and massive multiple-access techniques. He has served in various editorial roles, including as Area Editor for the IEEE Transactions on Communications, Guest Editor for the IEEE Network special issue on "Native Artificial Intelligence in Integrated Terrestrial and Non-Terrestrial Networks in 6G," and Guest Editor for the IEEE Open Journal of Vehicular Technology (OJVT) special issue on "Recent Advances in Security and Privacy for 6G Networks." Additionally, he has held positions as Senior Editor and Editor for IEEE Communications Letters, Editor for the IEEE Transactions on Communications, and Associate Editor for the IEEE Transactions on Vehicular Technology.

Authorized licensed use limited to: Khalifa University. Downloaded on April 20,2026 at 10:23:32 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted,
but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.