scientific reports

Check for updates

OPEN

DRLO-VANET: a deep reinforcement learning-based offloading framework for low-latency and energy-efficient task execution in VANETs

Sadineni Neelima[1]✉, S. Rama Sree[2] & N. Ramakrishnaiah[3]

Multi-access Edge Computing (MEC) is a key enabler for supporting latency-sensitive and compute-intensive applications in autonomous vehicles by integrating edge computing resources with vehicular Ad Hoc Networks (VANETs). Existing strategies for task execution, such as local-only, static offloading to RSUs, and greedy minimum-latency, exhibit critical flaws, including low scalability in dense traffic, imbalanced RSU utilisation, numerous handovers, and significant energy consumption. These challenges underscore the need for dynamic, intelligent multi-objective decision-making frameworks that enable real-time trade-offs. In this paper, we first identify challenging vehicular mobility and network conditions. Then we design DRLO-VANET, a framework for dynamic local execution vs. MEC offloading decision-making based on deep reinforcement learning. A novel framework for integrating NS-3 and ns3-gym, enabling online state-action-reward learning with sophisticated model-free DRL algorithms such as Deep Q-Networks (DQN) and Soft Actor-Critic (SAC), is presented. DRLO-VANET introduces a global state space that includes vehicular density, RSU load, task size, and channel quality, and jointly optimises latency, energy consumption, task completion ratio, RSU utilisation rate, and handover overhead. Through extensive simulations, experimental results reveal that DRLO-VANET decreases the task execution latency by up to 40%, saves energy (30%–35%), and enhances the task completion ratio to over 90% with medium density, and decreases the handover frequency by nearly 50% analysis compared to the baseline methods. As a result, these results verify that policies activated by DRL outperform other energy management strategies by balancing short-term responsiveness with long-term system stability. We proposed a scalable, adaptive, and efficient framework with real-time task offloading capabilities for vehicular tasks. Its performance was proven through initial experiments and simulations, making it suitable for new autonomous transport systems that require high reliability and responsiveness.

Keywords Vehicular Ad hoc networks, Multi-access edge computing (MEC), Task offloading, Deep reinforcement learning, Autonomous vehicles

VANETs are an essential component of next-generation intelligent transportation systems and autonomous driving applications. VANETs promote safety message communication by supporting real-time safety warnings, cooperative perception, and traffic management for dense urban and highway scenarios through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication within the broader Internet of Vehicles (IoV). Autonomous vehicles today constantly perform a multitude of rich, computationally and resource-intensive tasks, ranging from image recognition to multi-sensor fusion and trajectory prediction, that may be challenging to execute promptly on resource-constrained processors on board. This has made computation offloading an essential need, in which tasks are selectively offloaded to nearby edge servers to meet stringent latency and reliability requirements[1–4].

[1]Research Scholar, Department of CSE, UCEK, JNTUK Kakinada, India. [2]Department of CSE, Aditya University, Surampalem, A.P, India. [3]CSE Department, UCEK, JNTUK Kakinada, India. ✉email: neelima.sadineni@gmail.com

One such offloading mechanism could use multi-access Edge Computing (MEC), since it is natural to deploy compute and storage resources at roadside units (RSUs) and other edge nodes adjacent to moving vehicles[12,3,56–9]. MEC can significantly reduce end-to-end delay, backhaul congestion, and improve scalability for delay-sensitive vehicular applications (e.g., collision avoidance, HD map sharing, cooperative perception) compared with remote cloud data centres. The addition of MEC in VANETs enables vehicles to offload tasks to the nearest RSUs or process them locally based on channel conditions, RSU load, mobility state, and the urgency of the task at hand.

Nonetheless, making such offloading decisions is challenging in MEC-enabled VANETs, as the environment is highly variable and dynamic. Heuristic schemes like static offloading to the nearest RSU, or basic greedy techniques may optimise latency for specific settings, but fail to respond to changes in vehicle density, RSU overloading and task heterogeneity[10–17]. Processing only within the local environment is energy-efficient for smaller workloads, but it becomes impractical as task complexity increases. Current offloading frameworks based on traditional optimisation or finite-state models are limited to simple mobility models, do not model the interdependent trade-offs among latency, energy, and reliability, or optimise a single performance metric in isolation[6,18,19]. Thus, a scalable, adaptive decision-making mechanism is still required to learn to balance trade-offs among multiple conflicting objectives in realistic vehicular environments.

The second ingredient, about finding near-optimal offloading policies, deep reinforcement learning(DRL) has a great paradigm, as it offers an opportunity to learn the parameters of the offloading policy directly from environment interaction, without the need for closed-form network dynamics. A DRL agent can iteratively optimize its policy based on vehicle state observations (e.g. vehicle position, channel quality, RSU load, and task characteristics) to minimize long-term cost functions that are a weighted sum of delay, energy consumption, and task completion performance[4,17,20–23]. DRL is exceptionally well-suited to this application domain, especially for MEC-enabled VANETs, where the decision space includes local execution and multiple candidate RSUs, and the underlying environment changes dynamically over time.

This paper investigates the joint local/edge computation offloading problem for vehicular tasks with delay and energy constraints in an MEC-enabled VANET. Introduction: Given a naturally dynamic network state (vehicle mobility, wireless link quality, RSU server utilisation, and task deadline), it focuses on deciding how to execute locally or offloading each of the generated tasks to a specific RSU in a way that limits end-to-end execution latency and energy consumption, enable a high task completion ratio, RSU overload, and excessive handover.

To address this issue, this paper proposes DRLO-Vanet, a Deep Reinforcement Learning-based Offloading framework for VANETs to enable optimal vehicular computation distribution between onboard units and nearby MEC servers. Instead, DRLO-Vanetta jointly considers execution latency, energy consumption, task completion ratio, RSU utilisation, and handover overhead to develop a realistic DRL-based adaptive offloading policy that achieves low-latency, energy-efficient task execution in vehicular scenarios.

This study addresses the following research question: Can a deep reinforcement learning–based offloading policy, trained under realistic packet-level vehicular network dynamics, jointly optimise latency, energy consumption, task completion ratio, RSU utilisation, and handover overhead in MEC-enabled VANETs? To answer this, we propose DRLO-VANET, a DRL-driven offloading framework tightly integrated with NS-3 and ns3-gym. Unlike prior DRL-based offloading approaches that rely on abstract system models or static mobility assumptions, the proposed framework embeds real-time channel variations, RSU queueing behaviour, and mobility-induced handovers directly into the learning loop. This enables the DRL agent to learn policies that are not only performance-optimal but also robust and deployable under realistic VANET conditions. This research contributes in three main ways:

- We formulate vehicular task offloading in MEC-enabled VANETs as a multi-objective sequential decision problem that jointly optimises latency, energy consumption, task completion ratio, RSU utilisation, and handover overhead under realistic mobility and wireless dynamics.
- We design DRLO-VANET, a DRL-based offloading framework tightly integrated with NS-3 and ns3-gym, enabling packet-level modelling of channel conditions, RSU queues, and vehicular mobility during policy learning.
- We implement and evaluate DQN- and SAC-based offloading agents through extensive NS-3 simulations, demonstrating consistent improvements over static and greedy baselines across varying traffic densities and channel conditions.

The rest of the paper is structured as follows. Section “Preliminaries provides” background on the VANET offloading problem, related work on routing, and on solutions using DRL. Preliminaries are introduced in Sect. “Preliminaries provides”, including background on VANETs, MEC, and task offloading, as well as a brief overview of deep reinforcement learning for vehicular applications. The proposed methodology extends to the system architecture, problem formulation, the aDRL agent, the algorithmic workflow, and the diagrams presented in detail in Sect. “Proposed framework”. In Sect. “Experimental results”, we detail and discuss the simulation results, including simulation configurations, performance measures, comparisons against baselines, and an exhaustive analysis of latency, energy, task completion ratio, RSU usage, and handover overhead. In Sect. “Discussion”, we summarise the results, and in Sect. "Limitations of the Study", we outline the study’s limitations. Finally, in Sect. "Conclusion and future work", we conclude this paper and describe future work, including extensions to security, federated learning, and real-world deployment.

Related work

Research on VANETs has increasingly focused on integrating computation offloading, DRL, and edge intelligence to achieve low-latency and energy-efficient task execution. Existing studies span system evolution, offloading strategies, reinforcement learning frameworks, energy and delay optimisation, and secure, trust-aware models, providing the foundation for developing advanced offloading frameworks such as DRLO-VANET.

Evolution of VANETs and emerging AI paradigms

As a core component of intelligent transportation systems, VANETs play a significant role, as their integration with AI has revolutionised how we manage tasks, communicate, and secure systems. Salcedo Parra et al. Rosmaninho et al.[10] surveyed the history of VANETs from an AI and SDN perspective. Parmar et al.[10] studied orchestration in edge–cloud continuums in the context of smart city deployment. Ali et al.[11] have proposed a multi-objective secure offloading using DDQN in blockchain-enabled IoV-MEC systems. Secondly[18], ensure task authentication and offloading using two fuzzy methods with zero-trust security. ZSM standards have been used to strengthen security slice management in MEC-supported V2X networks[19], while deep-learning-based emergency message dissemination frameworks have been reported in[24].

Edge computing has also shaped VANETs, where Ibn-Khedher et al.[2] proposed AI-assisted next-generation driving frameworks. Surveys such as those by Ahmed et al.[3] classified vehicular task offloading strategies, and Lakhan et al.[5] explored blockchain-based offloading with mobility-awareness. Kamoi et al.[6] designed platoon-based offloading mechanisms, while UAV-assisted traffic management has been optimised through multi-objective simulated annealing[25,26]. Aizaz Ul Haq et al.[27] surveyed SDN-based vehicular edge computing, and Saad et al.[28] presented TD3-based edge server selection for industrial and C-ITS contexts. Baktayan et al.[29] provided a systematic mapping of UAV-enabled MEC offloading, while Xu et al.[7] combined digital twins with blockchain for secure task offloading. Adaptive reinforcement learning with heuristic integration has been used for fog resource allocation[30], while Ye et al.[20] proposed federated double DQN for mobility-aware clusters. Deng et al.[8] advanced edge collaboration methods for IoV, and Waheed et al.[9] consolidated computing paradigms for vehicular offloading. Reinforcement learning-based reviews for fog environments[21] have further underlined the trajectory toward AI-driven VANET task management.

Computation offloading strategies in vehicular networks

Task offloading has shifted from heuristic-based optimisation to AI-powered frameworks. Hejja et al. Network slicing with load balancing was utilised by[12], while Vemireddy and Rout[13] proposed a fuzzy RL for energy-aware offloading. Wang et al. have actually deployed DRL-based offloading[4]. and Lin et al. In the context of vehicle control performed automatically based on[22], Phung et al. and Wu et al.[14] proposed oneVFC, an IoV fog computing platform for connected and autonomous vehicle scenarios. DRL was used in[15]for hybrid delay-constrained offloading. Liu et al. Performance-aware Offloading: The offloading design has been improved by incorporating task dependency awareness[31]. A review of intelligent offloading in MEC was given in[16].

Zhu et al. Delay-aware platooning frameworks were examined in[32] and Chen et al. The work in[17] tackles seamless service migration in a multi-edge IoV. Sinthia et al. DRL and federated learning based cache optimisation[33–39]. Parvini et al. at the greater resource allocation level. They further highlighted that previous optimisation techniques are being replaced by ML-based solutions[40]. For example, Huang[41] studied multimedia task offloading based on graph neural networks, and Wang et al. Dependency-aware UAV-assisted frameworks based on generative AI have been developed[42]. Ji et al. GNN and DRL have been integrated for V2X allocation[43,44], and a survey of 6G V2X resource management is presented in[45]. Talebkhah et al. Contextualised task offloading in Industry 4.0 and beyond.

Reinforcement learning and deep RL for task offloading

Reinforcement learning has become a dominant paradigm for adaptive vehicular offloading. Fuzzy RL[13], DQN[4], and DRL for autonomous vehicles[22] illustrate the transition from classical models. Wu et al.[15] extended RL to hybrid delay-sensitive contexts, and Cheng et al.[30] applied adaptive DRL with heuristics. Ye et al.[20] introduced federated double DQN, while Moghaddasi et al.[11] integrated DDQN with blockchain. Saad et al.[28] advanced TD3-based offloading, and Li et al.[23] proposed secure multi-slot DRL-based frameworks.

Emerging directions include dependency-aware DRL[46], multi-hop DRL offloading[47], federated DRL for cross-domain optimisation[48], and MDP-based multi-time-scale offloading in aerial IoV[49]. Hao et al.[50] proposed fairness-guaranteed DRL, while Wu et al.[51] employed asynchronous federated DRL for cooperative caching. Multi-agent DRL has been used for transmission control[52] and V2E resource allocation[53]. Song et al.[48] extended DRL to cross-domain networks, while Sadatdiynov et al.[54] surveyed optimisation in edge offloading. Other advancements include STAR-RIS-assisted spectrum allocation with DRL[55], decentralised multi-agent DQN[56], and DRL for 6G-enabled environments[45,57]. Surveys by Liu et al.[58] and Chatterjee et al.[59] further categorised RL-based offloading approaches.

Energy efficiency, latency reduction, and resource optimisation

Energy efficiency and latency reduction are central challenges in vehicular networks. Mustafa et al.[25,26] proposed MOSA and CPFT-MOSA for UAV-assisted allocation, balancing energy, delay, and robustness. Deng et al.[8] used edge collaboration to reduce delays, while Liu et al.[31] incorporated task dependencies. Zhu et al.[32] examined platoon-based delay reduction, and Chen et al.[17] discussed seamless migration strategies. Sinthia et al.[33] optimised caching using DRL and mobility-awareness.

Zhang et al.[60,61] studied DRL for optimising AoI and energy consumption in C-V2X systems, while Elgendy et al.[62] used DRL to optimise energy efficiency in vehicular edge-cloud networks. Li[63] applied DRL to mobile edge computing, while Luo et al.[64] introduced self-learning algorithms for IoV offloading. Cross-domain

federated DRL[48] and fairness-aware allocation[50] further improved optimisation. Hsu et al.[65] addressed quality of service promotion, while Labriji et al.[66] studied MEC migration strategies. UAV-assisted offloading in 3D mobility contexts was proposed by Liao et al.[67]. Xu et al.[7] employed digital twins, while Zhang et al.[68] explored caching in social-aware vehicular networks. Maleki et al.[69] investigated machine learning-based mobility-aware offloading, demonstrating significant efficiency gains.

Security, trust, and future challenges in VANET offloading

VANET offload — Security and privacy. Ali et al. While[18] proposed dual fuzzy trust-aware authentication, Asensio-Garriga et al. DDoS mitigation via ZSM slicing was the subject of[19]. Lakhan et al. Secure Offloading Based on Blockchain is presented in[5,9], which block all offloading tasks on one or more blocks using permutations to make offloading as secure as possible and prevent attackers' attacks. Secure task allocation based on DRL[23]. Huang[41] and Wang et al. Alternatives such as those from[42] focus on multimedia and UAV-assisted task design and incorporate security-aware concepts. Parvini et al. ML-based allocation remains a future work direction according to[40,49], which modelled security-constrained service placement.

Li et al. DDoS Attack is an example of RL applied to DDoS mitigation[70] Lin et al. Li et al.[71] proposed combining blockchain with RL for medical IoV. Wu et al. Biede et al.[58] used federated DRL for caching while keeping data private, and Brehon–Grataloup et al. MEC in V2X was surveyed from a security perspective in[72]. Chatterjee et al. provide a survey of VANET routing, focusing on RL-based robustness in[59]. Noman et al. highlighted the role of machine learning in 6G security. Annu and Rajalakshmi[45] highlighted issues with resource allocation. Maleki et al. Privacy was also discussed in the context of mobility-awareness offloading in[69].

Recent contributions also focus on mobility, caching, and workload orchestration in vehicular edge environments. Anokye et al.[73] proposed DRL-based UAV content caching and placement strategies, while Dziyauddin et al.[74] presented a comprehensive survey of computation offloading and content delivery in vehicular edge networks. Xie et al.[75] introduced MOB-FL, a mobility-aware federated learning framework for connected vehicles, and Chaowei et al.[76] advanced collaborative caching assisted by cell-free massive MIMO. Safavifar et al.[77] enhanced VRU safety through mobility-aware workload orchestration using RL-based trajectory prediction, whereas Han et al.[78] addressed dynamic task offloading and service migration optimisation in edge networks. Jeon et al.[79] further developed mobility-aware optimal offloading strategies in distributed edge computing. Collectively, these works highlight mobility-awareness, federated intelligence, and orchestration as crucial directions for scalable and secure VANET offloading.

Prior work on vehicular task offloading has studied a broad range of schemes, from static offloading heuristics to intelligent reinforcement-learning-based decision-making. Static nearest-RSU offloading approaches[12,14,16,17] achieve lower latency during light traffic but incur significant RSU saturation and performance degradation during heavy traffic, due to their ignorance of instantaneous channel and queue behaviour. Fuzzy logic and rule-based methods[13,15] can offer partial adaptivity to network dynamics. Yet, they are still constrained by limited available state information and lack an explicit long-term performance optimisation framework. Although adaptive learning is introduced for connected and autonomous vehicle (CAV) scenarios[4,15,22,23], most existing DRL-based frameworks are limited to simplified mobility patterns and a narrow RSU selection space, or treat multiobjective problems (e.g., delay[4] or energy[15,22] as a single objective, restricting their usefulness in realistic VANETs. The previous work on multi-agent DRL[20,48,51] and DRL in federated environments[16,73,77,80] are helpful in that they scale better than the single agent cases in exchange for sacrificing the vehicle-centered real-world vehicle characteristics consideration such as the handover overhead, mobility-aware RSU coverage transitions and heterogeneous task deadlines. Likewise, existing edge-assisted offloading surveys[3,5,40,58,74] show that current solutions rarely jointly optimise latency, energy consumption, the balance of RSU utilisation, and the task completion ratio.

Thus, while there are some promising RL/DQN applications previously, they do not provide a complete offloading model that combines vehicular mobility, realistic RSU load, multi-dimensional objectives, and instantaneous NS-3–level channel dynamics — gaps that the proposed DRLO-Vanet framework explicitly addresses.

Preliminaries

In this section, we provide the required theoretical background behind the proposed framework. First, it gives an overview of VANETs, MEC, and task offloading scenarios, and their roles in supporting autonomous vehicular applications. Next, it describes deep reinforcement learning (DRL) concepts relevant to offloading decision-making, along with the basic system assumptions and principles that underlie the methodology and experiments discussed in the rest of the paper.

Background on VANETs, MEC, and task offloading

Vehicular Ad Hoc Networks (VANETs) are a class of special Mobile Ad Hoc Networks (MANETs) in which the nodes are vehicles with onboard communication units, providing high diversity and a high level of services to support V2V, V2I, or V2X communications. In V2V mode, data such as vehicle positions, velocities, and safety warnings is exchanged between vehicles to improve traffic safety and efficiency. In contrast, vehicle-to-infrastructure (V2I) involves interaction with fixed infrastructure (e.g., approaches, routes, and traffic management), including roadside units (RSUs) that provide computing support for routing and related services. V2X communication, on the other hand, generalises both of these concepts across pedestrian, network-connected devices, and the cloud/edge server. However, low-latency communication remains a challenge due to the inherent properties of VANETs, namely high mobility, rapid topology changes, and the high-efficiency nature of the vehicles.

MEC enables offloading of such computations and can relieve resource-constrained onboard units (OBUs) by allowing MEC-equipped RSUs to host applications such as real-time image analysis, navigation assistance, and cooperative perception. In this context, the end-to-end latency of an offloaded task comprises three components: the transmission delay from the vehicle to the RSU, the processing delay at the MEC server, and the reception delay when sending the result back to the vehicle. Let $L_{\mathrm{tx},t}$ denote the transmission delay at decision epoch t, $L_{\mathrm{proc},t}$ the MEC processing delay, and $L_{\mathrm{rx},t}$the reception delay. The total latency L_t of an offloaded task can then be decomposed as in Eq. (1):

$$L_t = L_{\mathrm{tx},t} + L_{\mathrm{proc},t} + L_{\mathrm{rx},t}. \tag{1}$$

MEC integration minimizes $L_{\mathrm{proc},t}$ through localized computation and reduces $L_{\mathrm{tx},t}$ and $L_{\mathrm{rx},t}$ because of shorter communication paths, leading to lower overall latency and better support for time-critical vehicular applications.

Task offloading in vehicular environments has recently gained new attention as a viable solution to overcome the limitations of OBU acquisition due to limited processing power and energy budgets. This enables high-complexity computational algorithms (e.g., DNN inference or 3D map reconstruction) to be performed with rigorous latency constraints by offloading processing to either MEC servers or nearby high-performance vehicles. Nevertheless, the high dynamism of VANETs gives rise to several critical issues, including volatile channel conditions, sporadic connectivity, varying RSU loads, and unforeseen vehicle mobility patterns. These characteristics have implications not only for the feasibility of the decision process to offload but also for its efficiency; therefore, static or heuristic-based strategies would fall short in such dynamic environments.

But for autonomous vehicles, dynamic, smart offloading transitions from an optimisation alternative to a non-negotiable necessity. Decisions for similar applications (i.e., cooperative perception, collision avoidance, and HD mapping) need to be made within a few milliseconds, often in the presence of uncertain link stability and resource availability 10. A novel offloading framework that adaptively decides before azes are locally processed, MEC offloaded, or hybrid executed over MEC to optimise at the same time both latency and energy cost, without losing any application reliability. This adaptability can be framed as a cost-function optimisation problem with associated weights as described in Eq. (2):

$$\min_{a_t} \alpha\, L_t + \beta\, E_t \tag{2}$$

subject to task deadline and network resource constraints, where a_t is the offloading decision at time step t, L_t denotes the end-to-end latency of the task at time t as defined in Eq. (1), and E_t represents the total energy consumption (computation and communication) for executing that task. This optimisation must be solved in a continuously changing VANET environment, which motivates the use of machine learning and deep reinforcement learning approaches capable of learning near-optimal policies by balancing exploration and exploitation during interaction with the environment.

As shown in Fig. 1, VANETs can communicate via V2V, V2I, or V2X, enabling autonomous cars to communicate not only with each other but also with MEC-enabled RSUs. RSUs are communication gateways and local computation nodes that do not bring tasks to cloud servers but push computation to vehicles, thereby reducing latency. As shown in the diagram, tasks are characterised by data size, CPU cycles, and deadline, and can be handled in the vehicle or offloaded to MEC servers. This also reflects that logical decision-making should be used to minimise latency and energy consumption while accounting for deadline constraints, particularly due to frequent changes in channel quality, RSU load, and vehicle speed in dynamic mobility scenarios. These adaptive and learning-based strategies are crucial for the real-time performance of autonomous vehicles, as they facilitate a sequential flow from state information gathering to offloading decision-making.

Deep reinforcement learning for offloading decisions

The improved Vehicle MEC modality of DRL has opened new avenues for making optimal offloading decisions in a highly dynamic vehicular MEC environment. Unlike conventional heuristic-based approaches, which are constrained by pre-defined static rules, DRL enables autonomous vehicles to continuously interact with the driving scenario, thereby learning the optimal policy. We model this interaction as MDP where at any given time step t the agent (vehicle) observes the system state S_t, takes an action a_t from the action space, gets some reward R_t and transits to new state S_{t+1}. As the agent receives rewards, it learns how to better adapt its decisions by exploiting the changing conditions of the network, the patterns of vehicle mobility and the availability of resources to maximize the expected cumulative reward.

The state space in vehicular offloading scenarios comprises of parameters including channel quality (RSSI, SINR), RSU queue length, available computation capacities, vehicle speed and location, battery status, and task-related parameters such as data size, CPU cycles, and deadline. A typical action space comprises discrete actions like full local execution, partial task offloading to one RSU, or a continuous action for partial task splitting with ratio r∈[0,1]. We thus formulate the problem as finding a policy π^* that optimally trades off latency and energy consumption such that application deadlines are met at minimum cost as in Eq. (3).

$$\pi^* = \arg\min_{\pi} E[\alpha\, L_t + \beta\, E_t + \gamma\, P_{miss}] \tag{3}$$

where α, β, γ are the weights, L_t is the latency for a task, E_t is the energy consumption, and P_{miss} is the penalty for violating a deadline.

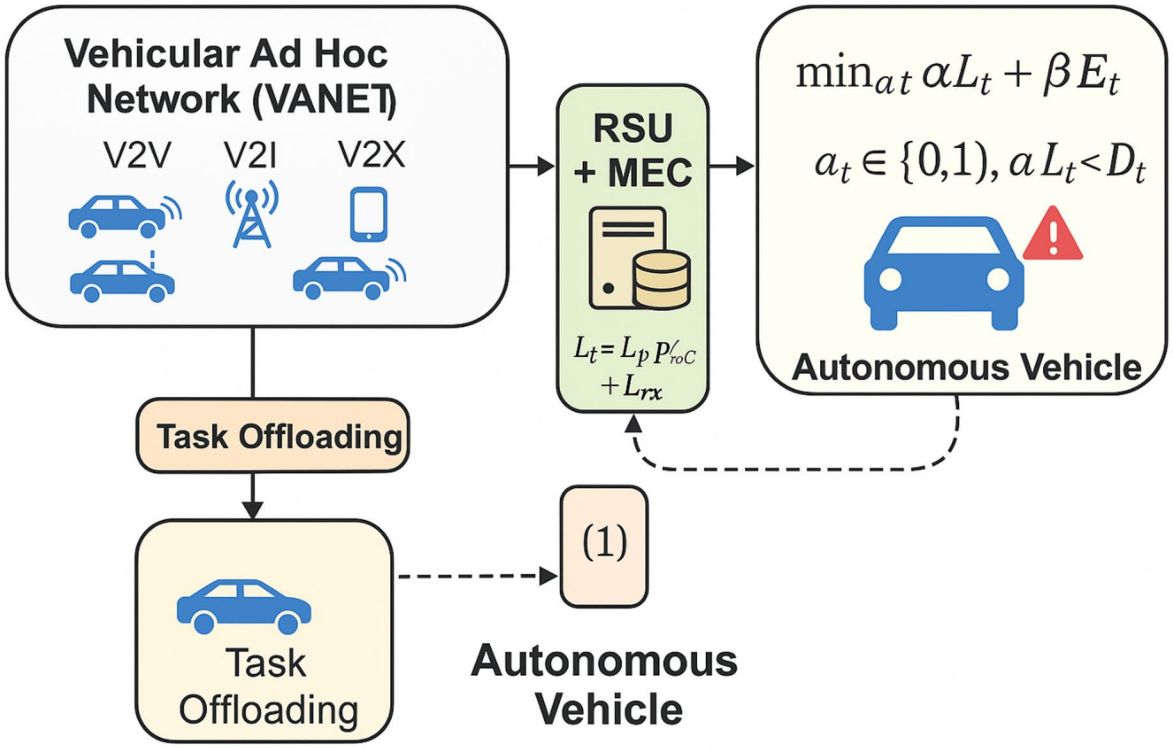

Fig. 1. Overview of VANETs, MEC, and task offloading for autonomous vehicles.

While discrete offloading decisions can be efficiently learned using DRL approaches such as Deep Q-Networks (DQN), SAC, or Deep Deterministic Policy Gradient (DDPG), continuous split ratios can be leveraged for hybrid offloading. DRL is incorporated to address adaptive scenarios involving variable wireless channels and high vehicular mobility in the context of offloading in a VANET environment. In addition, DRL agents can incorporate predictions, such as mobility predictions, to proactively determine offloading instances before connectivity deteriorates, thereby achieving higher task success rates and lower resource usage. Such adaptability is essential for autonomous vehicles executing safety-critical and low-latency applications.

Based on the onboard battery level of the autonomous vehicle, it decides whether to perform local execution or MEC offloading as a function 01 (5), therefore we state: · Battery Level: battery of the autonomous vehicle. (In the context of battery level, it only influences the decision-making whether to perform local execution or MEC offloading. However, the proposed DRL agent (DQN/SAC is implemented in this paper and runs in the NS-3 + ns-3-gym environment on a training machine outside of the vehicle. The Instant Policy is then executed by the car using the learnt model.

The DRL-based offloading framework in VANETs is detailed in Fig. 2. The NS-3 simulation environment is the core foundation, where autonomous vehicles communicate via V2I links with RSUs that include MEC servers. System state information for a given vehicle, such as channel quality, vehicle mobility, load on RSUs, battery level, and task-specific information, is measured periodically by each car. This state vector is what is given to the policy network for both DQN and SAC implementations, optimised across these parameters.

Given the observed state, the DRL agent determines the appropriate action—either executing the task locally on the vehicle-embedded onboard unit or offloading it to a selected MEC server. The decision is made, and the task is executed with the chosen execution path, and the performance impacts, including the latency L, energy consumption E, and deadline penalty Pmiss, are calculated. These metrics are used to compute the reward that is cascaded back to the DRL agent through reinforcement learning to update the policy. Such bounded interaction allows the agent to learn optimal offloading policies over time and adapt to network dynamics and mobility conditions, thereby minimising task latency and energy consumption. Notation used in the proposed methodology: (Table 1)

Proposed framework

We elaborate on the proposed DRLO-VANET framework in this section. It starts by mathematically formulating the offloading optimisation problem and the system architecture. The following subsections detail the simulation environment, DRL agent design, state→action→reward mapping, and algorithm workflow, with appropriate figures and tables. Collectively, these three elements form the methodological basis for assessing the frameworkis efficacy.

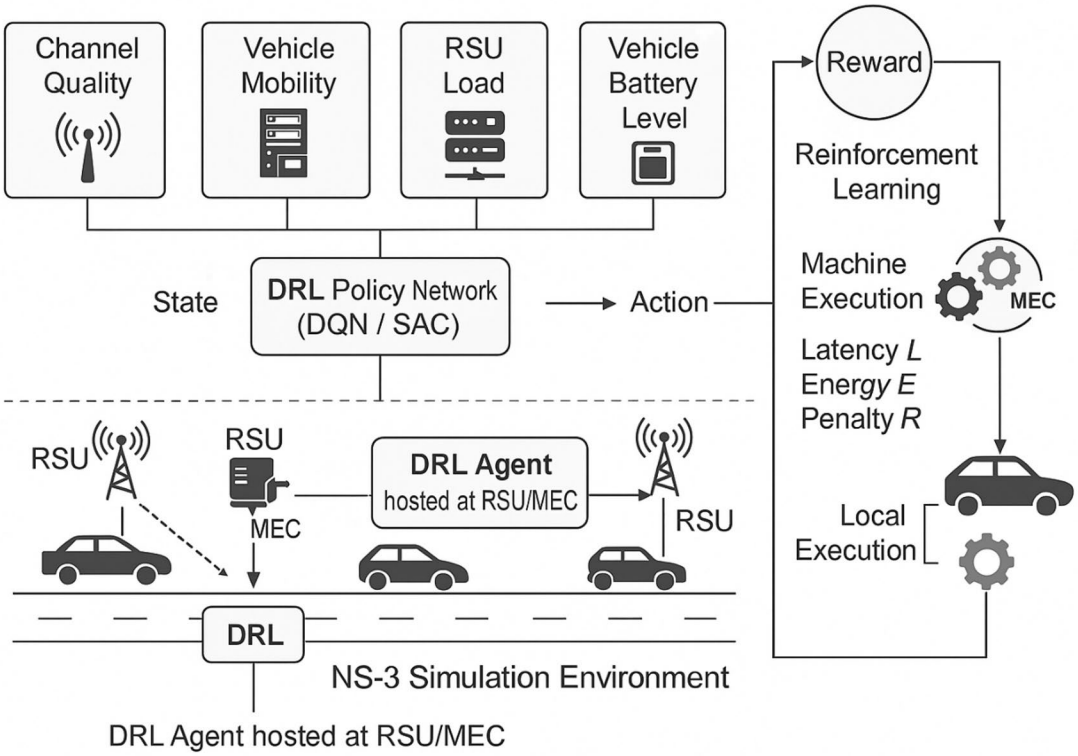

Fig. 2. DRL-based offloading framework in VANETs.

We consider a discrete-time decision process indexed by decision epochs t. At each epoch, a vehicle generates a computation task characterized by input data size S_t, required CPU cycles C_t, and deadline D_t. The system state s_t captures the network, mobility, resource, and task context, while the action a_t represents the offloading decision. Actions include local execution, offloading to a selected MEC-enabled RSU, or hybrid execution via a split ratio. The objective is to learn a policy $\pi\ (s_t)$ that optimizes latency, energy consumption, task completion ratio, and handover overhead under dynamic VANET conditions.

Problem definition

The task offloading decision in the DRLO-VANET framework is modelled as a sequential decision problem in which each autonomous vehicle must decide whether to execute the task locally or offload it to one of the MEC-enabled RSUs. At time t, a new task T_i is generated with input data size D_i, required CPU cycles C_i, and execution deadline DL_i. The discrete decision variable $a_t \in \{0,1,\ldots,K\}$ encodes the offloading choice, where $a_t = 0$ denotes local execution on the vehicle OBU and $a_t = k$ (for $k \in \{1,\ldots,K\}$) represents full offloading to RSU k. When hybrid execution is enabled, we additionally introduce a continuous split variable $r_t \in (0,1)$ indicating the fraction of the task processed at the MEC server (with the remaining $1 - r_t$ executed locally). The optimization objective is then to find a policy π^* that minimizes a weighted sum of task latency and energy consumption while maximizing the task completion ratio under dynamic VANET conditions, as expressed in Eq. (4).

$$\pi^* = \underset{\pi}{\operatorname{argmin}} E[\alpha\, L_t + \beta\, E_t - \delta\, R_t^{comp}] \tag{4}$$

In this equation, L_t is the total latency of task execution, E_t is the energy cost, R_t^{comp} is the ratio of the number of completed tasks, and α, β, and δ are weights that show the importance of each performance parameter relative to one another.

This problem formulation should adapt to multiple real-world constraints from the practical aspects of a VANET domain. Link quality parameters such as sinr, rssi, fluctuate due to network dynamics (specifically the fast movement of vehicles and interference caused by environment), which translates into direct impact of increased transmission delays and loss of packets. The amount of resource available for processing in an RSU is limited — each MEC server has a finite computing capability C_{RSU} and the queue length $Q_{\max}$ the number of tasks it can process at one time, is also finite. This means that every task has to be executed within certain task deadline, and sum of transmission, processing and reception delays needs to comply with $L_i \leq \mathrm{D}\ L_i$. Additionally, they make the process of deciding more complex because vehicle trajectory affected by the urban mobility model leads to frequent RSU handovers, which may break the ongoing offloading session.

Notation	Description
$T_i\,(t)$	Task generated by vehicle $i$at decision epoch t, defined by $\langle D_i, C_i, DL_i \rangle$
D_i	Input data size of task i(bits)
C_i	Required CPU cycles for task i
DL_i	Execution deadline for task i(ms)
a_t	Offloading action selected at decision epoch t(0: local execution; k: offload to RSU k; $r \in [0,1]$: task split ratio)
L_t	End-to-end latency of the task executed at epoch t(s)
E_t	Total energy consumed for task execution at epoch t(J)
R_t^{comp}	Binary task completion indicator at epoch t(1 if completed within deadline, else 0)
$\alpha\,,\beta\,,\gamma\,,\delta\,,\nu$	Weighting coefficients in the optimization objective and reward function
L_{tx}	Uplink transmission delay for task offloading (s)
L_{proc}	Processing delay at the MEC server or OBU (s)
L_{rx}	Downlink reception delay for task results (s)
f_{MEC}	CPU processing rate of the MEC server (cycles/s)
f_{OBU}	CPU processing rate of the onboard unit (cycles/s)
$T_{\text{proc}}^{\text{MEC}}$	Processing time of task at the MEC server (s)
$T_{\text{proc}}^{\text{loc}}$	Processing time of task executed locally on the OBU (s)
R_{link}	Achievable wireless link data rate (bits/s)
W	Channel bandwidth (Hz)
SINR_{eff}	Effective signal-to-interference-plus-noise ratio
P_{CPU}	Power consumption of the onboard CPU (W)
P_{TX}	Power consumption of the wireless transmission module (W)
E_i^{comp}	Energy consumed for local computation of task i(J)
E_i^{tx}	Energy consumed for transmitting task $i$to MEC (J)
TCR	Average task completion ratio over the observation window
B_j	Total busy time of the MEC server at RSU j(s)
p_j	Utilization ratio of the MEC server at RSU j
h_v	Number of RSU handovers experienced by vehicle v
$H^{'}$	Mean number of handovers per vehicle
T_{obs}	Total observation period for performance evaluation (s)
n_v	Vehicle density (vehicles per km ^{2})
$G = (g_x, g_y)$	RSU grid dimensions along horizontal and vertical axes
d_{RSU}	Distance between adjacent RSUs (m)
R_{cov}	Coverage radius of an RSU (m)
Ω	Mean RSU overlap factor (average number of RSUs within vehicle coverage)

Table 1. Notations used in the proposed methodology.

Here, the performance objectives of the DRLO-Vanet agent are to maximize the task completion ratio and execution efficiency while minimizing the overall task latency and energy consumption. Reducing latency of real-time applications ensures timely response, energy optimization prolongs vehicle operational endurance by balancing computation and transmission energy consumption, and maximizing task completion ratio improves confidence over reliability of safety-critical autonomous driving functions. Consequently, DRLO-VANET uses deep reinforcement learning to obtain adaptive, context-aware policies that can work with stochastic variations (e.g., randomness in network conditions, vehicle mobility, and MEC server loads) and achieve nearly-optimal task offloading decisions without requiring a priori knowledge of the environment.

In this work, the energy consumption for both local computation and task offloading follows a CPU-cycle and transmit-power based formulation consistent with established MEC offloading models in vehicular networks[81]. For local execution, the energy required to process a task T_i with C_i CPU cycles is modeled as $E_{\text{comp}} = \kappa\, C_i$, where κ denotes the effective energy per CPU cycle. For offloading, the transmission energy is determined by the transmit power P_{tx} and the uplink transmission duration τ_{tx}, yielding $E_{\text{tx}} = P_{\text{tx}}\tau_{\text{tx}}$. Similarly, the downlink reception energy is expressed as $E_{\text{rx}} = P_{\text{rx}}\tau_{\text{rx}}$. Thus, the total offloading energy becomes $E_{\text{off}} = P_{\text{tx}}\tau_{\text{tx}} + P_{\text{rx}}\tau_{\text{rx}}$. This formulation provides a unified and reproducible foundation for evaluating the energy trade-off between local and MEC-assisted execution in dynamic VANET scenarios.

The transmission power P_{tx} represents the instantaneous energy expenditure of a vehicle when offloading a task to an MEC server. It is determined by the required data-rate and the wireless channel condition. Specifically, to achieve an uplink rate R_{ul} over a channel with bandwidth B and channel gain h, the minimum transmission power can be expressed using the Shannon capacity relationship as $P_{tx} = \frac{(2^{R_{ul}/B}-1)\,N_0 B}{h}$, where N_0 denotes the noise power spectral density. Since the uplink transmission energy is proportional to both the power and the transmission duration, the offloading energy cost becomes $E_{tx} = P_{tx} \cdot \tau_{tx}$, where τ_{tx} is the time required to upload the input data to the MEC server. This formulation ensures that higher required data rates or poorer channel conditions naturally lead to higher transmission energy usage, consistent with established MEC offloading models such as those adopted in[81].

System architecture

The DRLO-VANET system architecture, presented in Fig. 3, integrates autonomous vehicles, MEC-enabled RSUs, and a DRL decision-making agent within a unified NS-3 simulation environment. Vehicles are equipped with OBUs that handle sensing, communication, and local computation. These vehicles communicate with RSUs via IEEE 802.11p-based V2I links, while inter-vehicle communication is supported through V2V channels for cooperative data exchange. RSUs are strategically deployed along the roadside and equipped with MEC servers that provide localized computation and storage resources. These servers reduce task execution delays by processing offloaded workloads closer to the vehicles, avoiding the high latency associated with distant cloud servers.

The DRL agent, where the decision-making intelligence is being implemented, is external and easily interfaced with NS-3 via the ns3-gym API. The state observed by the DRL agent in the simulation environment consists of link quality indicators, moving vehicles provided by the simulation, queue lengths of the MEC Server, onboard resources, and task information. Using this information, the agent decides whether to execute the task locally on the vehicle, completely offload it to a chosen MEC server, or partially process it locally and remotely. The selected action is then fed back to NS-3, which simulates its execution and tracks performance metrics (i.e., task latency, energy consumption, and deadline fulfilment).

The architectures operate on discrete time slots — each time slot corresponds to a decision epoch. Vehicles produce tasks at each epoch, and the DRL agent interacts to find the optimal execution strategy. Subsequently, simulation-measured performance is fed back to the DRL agent as reward, and the DRL agent modifies its policy through the feedback policy to adapt to dynamic changes in network topology, channel conditions, and MEC Load. The TAR method is based on a closed-loop interaction strategy between MTC tasks and offloading. Thus, offloading decisions are made in real time and optimised to minimise both latency and energy consumption while guaranteeing a high task completion ratio in a realistic vehicular networking environment.

Simulation environment and network model

The DRLO-VANET simulation environment, listed in Table 2, is entirely implemented in NS-3[82] for high reproducibility of results and complete control over networking, mobility, and computation parameters. The

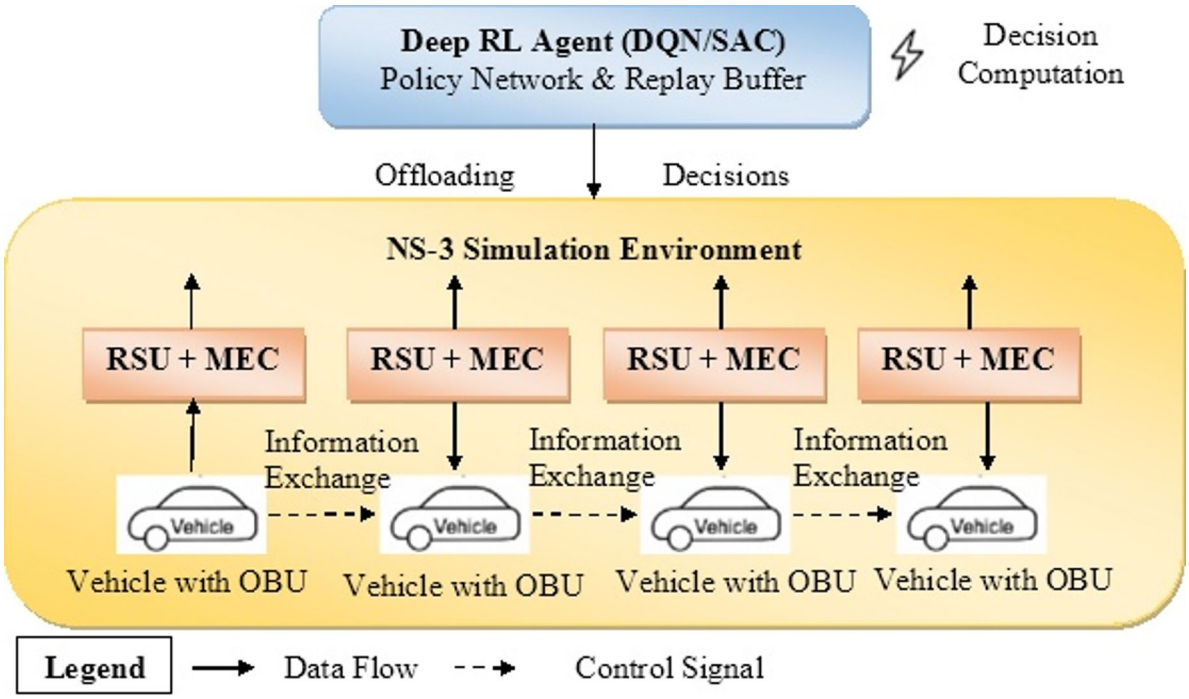

Fig. 3. System architecture of DRLO-VANET with Vehicles, RSUs, MEC, and DRL offloading loop.

Component	Description
Simulation Platform	NS-3 with ns3-gym interface for DRL integration
Communication Standard	IEEE 802.11p/WAVE for V2I and V2V communication
PHY/MAC Configuration	YansWifiPhy with Nakagami-m or log-distance path loss; QosWaveMacHelper for quality of service
Channel Parameters	10 MHz channel bandwidth, tuned Tx power, receiver sensitivity, and realistic MCS selection
Urban Grid Topology	Structured urban grid with intersecting roads
RSU Placement	RSUs are deployed at regular intervals to ensure overlapping coverage.
Vehicle Density	Scenarios from sparse (low traffic) to dense (urban rush-hour)
Mobility Models	Waypoint Mobility Model or Gauss-Markov for realistic movement patterns
MEC Server Emulation	Processing queue in NS-3 representing MEC computational capacity
Task Processing Model	Service time proportional to CPU cycles; first-come-first-served (FCFS) queue discipline
Energy Consumption Model	Models for local CPU usage and transmission-based offloading

Table 2. NS-3 simulation environment and network model parameters.

NS-3 design includes nodes corresponding to autonomous vehicles with IEEE 802.11p/WAVE network interfaces for V2I and V2V communication, roadside units (RSUs) with MEC Servers, and the ns3-gym interface[83] that establishes the link between the simulation and the external deep reinforcement learning agent. This simulation runs in discrete time, with each time step corresponding to a decision epoch during which vehicles create tasks for computation and access the DRL agent to find the best execution strategies.

You can obtain the IEEE 802.11p/W[1] AVE communication model in a realistic vehicular networking setup. For urban propagation characteristics, the PHY layer uses the YansWifiPhy module with Nakagami-m or log-distance path-loss models. The MAC layer, on the other hand, sits on top of QosWaveMacHelper (#QosWaveMacHelper) to enable high-priority message throughput. The channel bandwidth is 10 MHz, and MCSs are used to maintain a balance between high data rate and reliability, with the grid-average SINR varying from 0 to 30 dB. Tuning transmission power levels, receiver sensitivity, and special beacon intervals to meet vehicular networking standards ensures stable connectivity for both safety-critical and backhaul-offloading data.

The simulation topology is an urban grid, with roads at 90 degrees to each other and RSUs dispersed at equal intervals to avoid overlapping coverage zones and reduce communication dead zones. Use Case 1: Vehicle density ranges from sparse (low traffic) to dense (urban rush-hour conditions) to assess the impact of workload changes on the DRLO-VANET performance. The general placement of RSUs is optimised to achieve a trade-off between coverage efficiency and the utilisation of MEC servers, allowing vehicles to be served by multiple offloading options at most times.

Mobility can be either a Waypoint Mobility Model, in which vehicles have predetermined paths and speeds, with speeds and pauses at intersections selected randomly, or a Gauss-Markov Mobility Model, which implements speed and direction correlations to produce more realistic traffic flow patterns. These models, including link durations, handover events, and, in particular, the stochastic movement of vehicles in an urban environment, are then captured.

In NS-3, the behaviour of the MEC server is emulated as a processing queue; that is, each offloaded task is assigned a service time that can be computed based on the CPU cycles required to process it and the MEC server's computational capacity. Tasks are processed in a first-come-first-served manner, and queue lengths are periodically measured to reflect load information to the DRL agent. The proposed task processing model accounts for the summation of transmission delay to the MEC server and the processing delay while in the server's queue, so the execution time indeed corresponds to the actual MEC scenario. It provides models of energy consumption for both local execution on OBUs and transmission-based offloading, enabling an accurate assessment of the latency–energy trade-offs of DRLO-VANET.

DRLO-VANET offloading workflow

The DRLO-VANET offloading workflow, shown in Fig. 4, proceeds in discrete decision epochs synchronised with the NS-3 event scheduler. At the beginning of each epoch, every vehicle generates a computational task characterised by input size, required CPU cycles, and a deadline. It simultaneously samples the current network and resource context. This context includes PHY/MAC indicators such as RSSI/SINR to nearby RSUs, queue length and estimated service time at MEC servers, onboard CPU utilisation and battery state, vehicle speed and position, and task attributes. The collected measurements form the state vector provided to the deep reinforcement learning policy. The DRL agent performs inference to select an action that specifies local execution, full offloading to a particular RSU, or hybrid execution via a split ratio. NS-3 then executes the chosen action: local tasks are served by an emulated OBU CPU model, while offloaded tasks are packetized, transmitted over IEEE 802.11p to the target RSU, enqueued at the MEC server emulator, processed, and returned to the vehicle. Upon completion (or deadline expiry), NS-3 computes latency, energy, and deadline adherence and returns these as feedback to the learning loop, which converts them into a scalar reward and updates the policy during training phases.

Decision-making is tightly integrated into NS-3 time slots to maintain causality and real-time consistency. Each slot encapsulates the sequence of state sampling, action dispatch, and execution progress, with packet transmissions, queueing, and processing advanced by the simulator's discrete-event clock. This alignment ensures that handovers, interference, and contention naturally influence the subsequent state the agent observes,

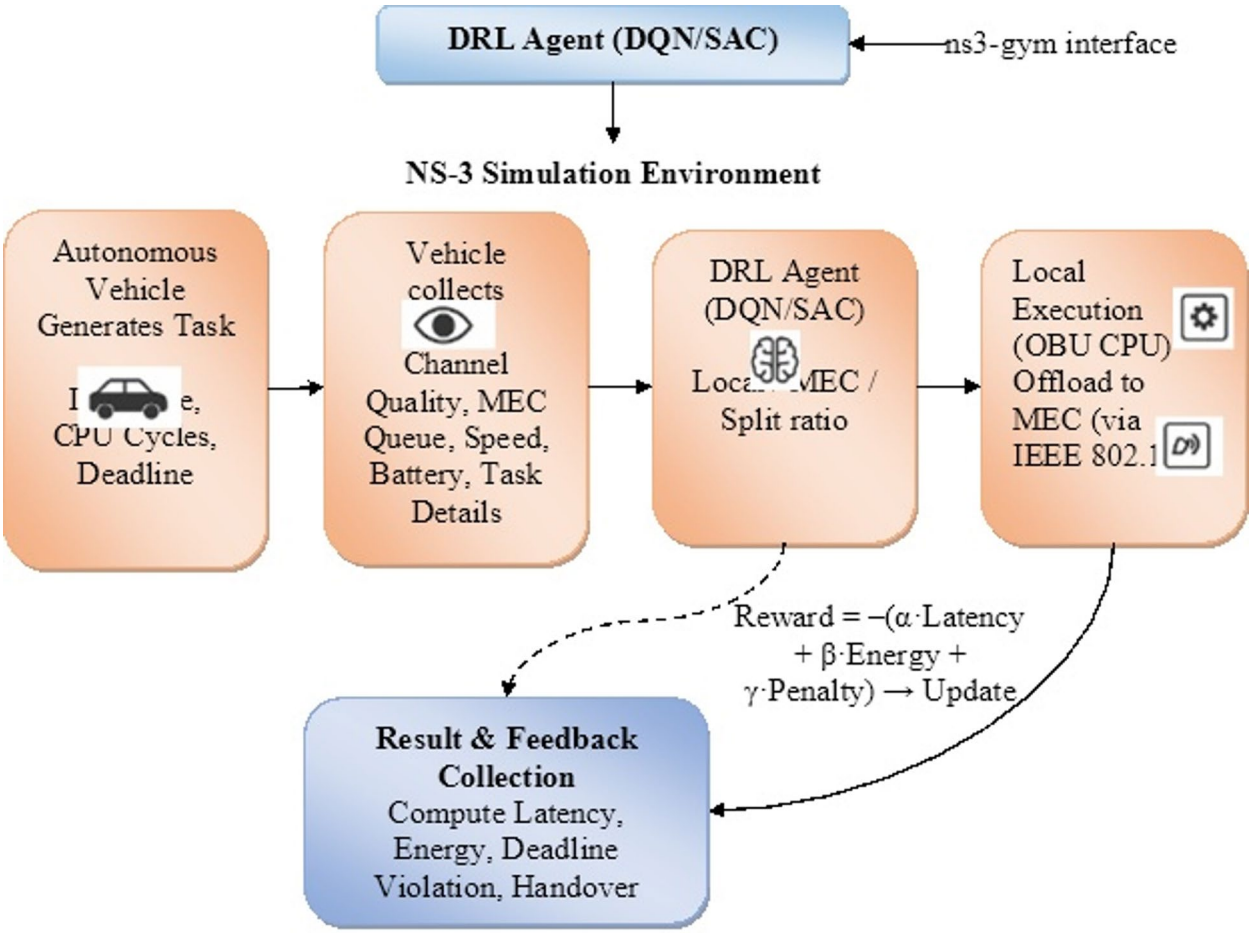

Fig. 4. End-to-end offloading workflow of DRLO-VANET with task generation, DRL decisions, execution, and iterative updates.

thereby reflecting actual protocol dynamics. For training, experience tuples (state, action, reward, next state) are accumulated per slot; for evaluation, the same loop runs with the policy frozen, ensuring that performance metrics are collected under identical timing semantics.

The reward function of the DRL consists of a combination of latency, energy consumption, penalty for missing deadline andpenalty for handover overhead in the form of a single scalar signal. Normalization of each component for numerical stability and balance in learning This formulation is written in a compact form and does not repeat intermediate derivations, as this is standard practice in multi-objective DRL based offloading. The DQN and SAC agents are trained with standard temporaldifference and entropy-regularized policy optimization mechanisms respectively. These update rules are very well-known in DRL literature, so we present their role conceptually and skip the derivations that are in the literature for brevity.

The ns3-gym API is the bidirectional interface that connects NS-3 to the Python-based DRL stack. NS-3 publishes serialised observations to the gym environment every epoch boundary, which then sets up the Python agent to read in for forward passes through the DQN/SAC network. The action output is returned to NS-3 via the same link and directly applied to configuration execution (e.g., RSU selection, split ratio application). When the simulation is advanced in NS-3 and outcome metrics are calculated, the reward and next-state are sent back to Python, where the agent records the transition into a replay buffer and, if in training mode, performs gradient updates before the next epoch. This low-latency exchange enables real-time co-simulation while maintaining the single-simulator design, ensuring reproducibility and enabling scaling to learn.

DRL agent design

At every NS-3 decision epoch, as presented in Fig. 5, the DRLO-VANET agent observes a compact state vector that fuses network and mobility, task, and resource hints. Link quality to candidate RSUs (e.g., RSSI/SINR and packet error rate in the recent past) and load indicators, such as the current queue length and estimated service time per RSU, are instantaneously aggregated into the network slice. The mobility slice contains vehicle speed and position indices for planning handovers, while the resource slice includes onboard CPU utilisation and battery level. The job characteristics, such as input size, CPU cycles required, and deadline, are encoded in the task slice to help the agent determine if a job can run locally within a time & energy bound. All features are normalised

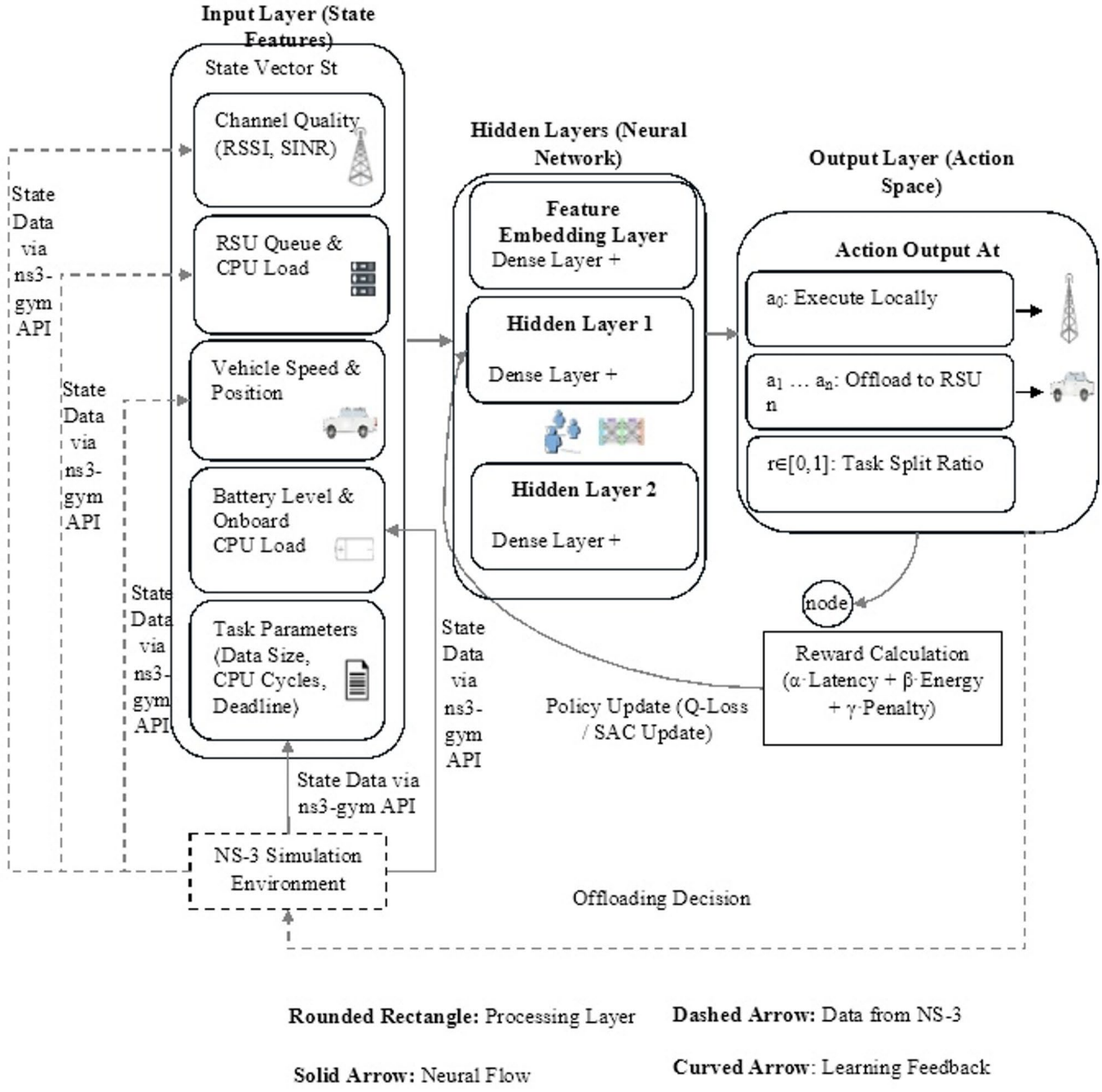

Fig. 5. DRL agent architecture of DRLO-VANET with state inputs, neural layers, action outputs, and reward-based policy updates via ns3-gym.

online for stable learning and to equalise magnitudes across dimensions, making them commensurate despite heterogeneous magnitudes.

The action emitted by the policy, which states how each task is performed in that epoch. In the discrete mode, the agent is given the option to perform locally on the OBU or full offload to one of the indexed RSUs. A split ratio r is produced by the policy in a continuous form r∈[0,1], which gives a slice of offloaded versus locally processed computation workloads, which translates to an intelligent hybrid execution amidst ephemeral link and queue conditions.

This reward term combines latency, energy, and reliability into a single scalar reward. Using the latency decomposition, which is previously defined in Eq. (4), the local processing time for on-device execution, or (3) for offloaded tasks, the per-epoch reward is given as in Eq. (5).

$$R_t = -\left(\alpha \widetilde{L}_t + \beta \widetilde{E}_t + \gamma \, 1\{L_t > DL_t\} + v\widetilde{H}_t\right) \tag{5}$$

with $\widetilde{L}_t$, $\widetilde{E}_t$, and $\widetilde{H}_t$ latency, energy, and handover costs normalized to [0,1], 1{·} penalizes deadline misses, and α, β, γ, ν weight the operational priorities. Normalization applies running min–max statistics to keep gradients in check as expressed in Eq. (6).

$$\widetilde{X}_t = \frac{X_t - X_{min}}{X_{max} - X_{min} + \in} \tag{6}$$

individually for each cost component X∈{L, E,H}. Our agent wants to return a discounted return function $J(\pi) = E_\pi \left[\sum_{k-0}^{\infty} \zeta^k R_{t+k}\right]$ with discount ζ∈(0,1).

Although the mathematical formulation primarily expresses the state vector in compact notation, the complete state representation used by the DRLO-VANET agent explicitly includes **vehicle speed** and **vehicle position** as part of the mobility-awareness component. These two variables are essential because vehicle mobility directly affects channel quality, link duration, task transmission time, and the likelihood of RSU handover. In the implementation, the global state s_t defined in Eq. (6) is extended as

$$s_t = \{q_t^{RSU},\ h_t,\ L_t^{queue},\ D_i,\ C_i,\ DL_i,\ v_t,\ p_t\},$$

where v_t represents the instantaneous vehicle speed and p_t denotes the vehicle's current position coordinates. These mobility parameters are fed into the DRL network to ensure that the offloading policy adapts to changes in connectivity, RSU coverage boundaries, and dynamic channel variations. This explanation aligns with the inputs shown in Fig. 6 and clarifies how vehicle speed and position influence the decision-making process.

In the case of discrete actions (local and RSU k), DRLO-VANET uses a Deep Q-Network to minimize the temporal-difference loss as in Eq. (7).

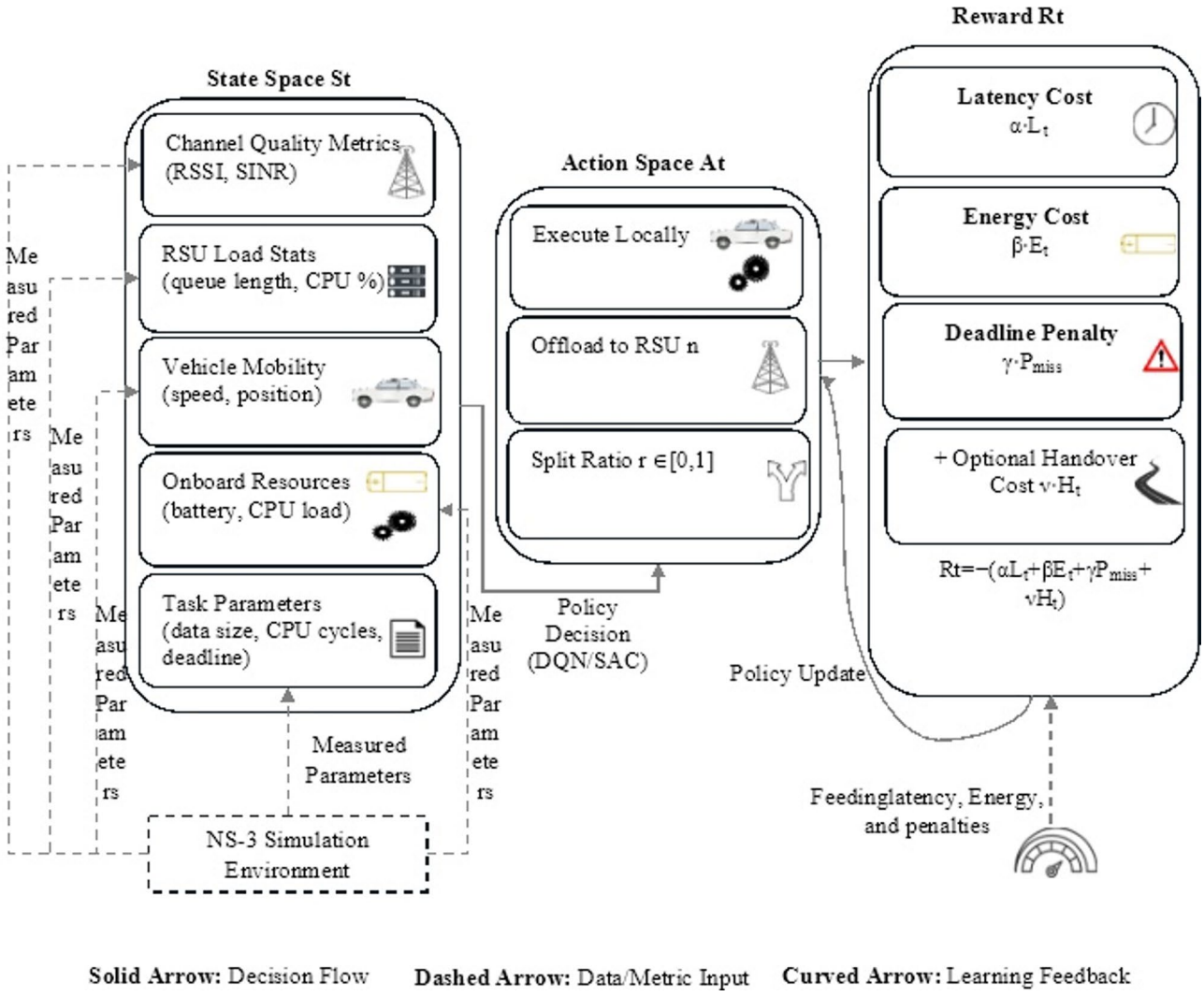

Fig. 6. State–action–reward mapping of DRLO-VANET with NS-3 features, offloading decisions, and reward calculation.

$$L_{DQN} = E\left[\left(R_t + \zeta \max_{a'} Q_{\theta^-}\left(S_{t+1},\ a'\right) - Q_\theta\ (S_t,\ a_t)\right)^2\right], \tag{7}$$

with target network θ^- and experience replay to decorrelate samples For continuous split-ratio control, the agent employs Soft Actor-Critic[41], optimizing a stochastic policy by maximizing an entropy-regularized objective J(π)= $E_{S_t\sim D, a_t\sim\pi}\ [Q_\varphi\ (S_t,\ a_t) - \eta \log\pi\ (a_t|S_t)]$ with respect to a critic fitted to a soft Bellman backup, yielding stable learning amidst rapidly changing VANET dynamics.

Exploration–exploitation is arranged to mimic the preparation stage and traffic administration. In DQN, the ε -greedy schedule starts from high exploration to expose the agent to a diverse set of link and load conditions and decays ε as the Q-function converges with a small floor such that the agent remains adaptive to topology changes. In SAC, the entropy temperature η is tuned(or learn)to utility a pre-defined action entropy, which keeps enough exploration on the split ratios when channels or RSU queues varies. In both scenarios, the replay buffer emphasizes rare but informative transitions like deadline violations and handovers ensuring that the policy learns to internalize the cost of unreliable links and congested MEC servers and finally tends to take actions that reduce normalized latency and energy, while respecting deadline.

The DRL agent is trained externally using the ns-3-gym interface, where NS-3 provides real-time environment feedback and the DQN/SAC model is executed on the training machine. Vehicles do not run the DRL model internally; instead, they follow the learned policy that maps the state to the offloading decision.

The hyperparameters of the DRL agents, including learning rate, discount factor, batch size, replay buffer size, and target network update rate, are selected based on established practices in off-policy deep reinforcement learning and prior MEC offloading studies. Initial pilot experiments were conducted on a validation scenario to ensure stable convergence and to avoid overfitting to any specific traffic or channel configuration. The same hyperparameter settings are then retained across all evaluated scenarios to maintain consistency and fairness in performance comparison.

State–action–reward mapping

In DRLO-VANET, the NS-3 simulation continuously exposes raw measurements that are transformed into the state variables consumed by the DRL policy at each decision epoch. PHY/MAC traces provide per-link RSSI/SINR and recent packet error rates to candidate RSUs; these are normalized and inserted into the network slice of the state. The MEC emulator exports instantaneous queue length and an estimated service time derived from its current workload and configured CPU capacity, which populate the load features. The mobility slice is built from the vehicle's current speed and position indices, along with a short history to capture trends relevant to impending handovers. Onboard resource monitors supply CPU utilization and battery level, while the task generator attaches the job's input size, required CPU cycles, and deadline. Together, these fields form a compact, normalized state vector aligned with the feature definitions in Sect. 4.5, ensuring that heterogeneous quantities appear on comparable scales as in Eq. (6). Figure 6 illustrates how NS-3 simulation features map into DRLO-VANET states, guide offloading decisions, and compute rewards for learning.

Simulation events determine actions emitted by the policy map. A local-execution action configures the OBU service module to enqueue the task into the on-device processor, advancing the discrete-event clock through the compute service without initiating any transmissions. An offload-to-RSU-k action instantiates a packet flow over IEEE 802.11p toward the selected RSU, where the task is enqueued at the MEC server emulator and processed according to its queue discipline and service rate before results are returned to the vehicle. When the policy supplies a split ratio, NS-3 proportionally divides input bytes or CPU cycles into local and remote sub-jobs and schedules both pipelines concurrently, with completion time determined by the slower branch. In all cases, handovers, contention, and interference are naturally reflected because link-layer transmissions, retries, and RSU coverage are governed by the simulator's channel and mobility models.

Rewards are computed from simulation-measured metrics and then combined using the formulation in Eq. (5). End-to-end latency is assembled from the components produced by NS-3 (e.g., transmission and reception timing, queuing and processing delays), consistent with the decomposition given in Eq. (1). Energy consumption aggregates the modeled on-device compute energy for local service and the radio transmission energy for offloaded bytes, with both streams normalized as in Eq. (6). A deadline-miss indicator is set if the measured latency exceeds the task's deadline, and an additional normalized cost reflects handovers incurred during execution. The resulting scalar reward penalizes large latency and energy while adding discrete penalties for unreliability, providing a stable learning signal that drives the policy toward actions that meet deadlines with minimal resource expenditure under realistic VANET dynamics.

Algorithmic workflow

Each NS-3 decision epoch is followed by DRLO-VANET loop execution. This is how a vehicle produces the current state (simulation outputs link quality to RSUs, MEC queue/load, speed/position, OBU CPU/battery, and task ⟨D,C,DL⟩ queries the policy and implements local execution, complete offload to a particular RSU or hybrid execution with a split ratio. Additionally, NS-3 takes care of transmissions, queueing, and processing; when completed (or the deadline expired), latency, energy, and deadline status are extracted and a scalar reward is calculated (as per Eq. (5), the next state is generated the transition is then stored and leveraged to update the parameters during training, while the same loop runs for inference with a frozen policy.

During training, the DRL agent maintains a parametric state–action value function $Q_\theta\,(s,a)$. For the DQN variant, the Q-function is updated using the standard temporal-difference rule. Given a transition (s_t, a_t, r_t, s_{t+1})sampled from the replay buffer, the target value $\widehat{y}_t$and Q-update are expressed in Eqs. (8) and (9).

$$\widehat{y}_t = r_t + \zeta \max_{a'} Q_{\theta^-}(s_{t+1}, a'), \tag{8}$$

$$Q_\theta(s_t, a_t) \leftarrow Q_\theta(s_t, a_t) + \eta\left[\widehat{y}_t - Q_\theta(s_t, a_t)\right], \tag{9}$$

where $\zeta \in (0,1)$ is the discount factor, $\eta > 0$ is the learning rate, and θ^- denotes the parameters of the slowly updated target network.

For the SAC variant, we use two critic networks Q_{ϕ_1} and Q_{ϕ_2}. The soft Q-target for a sampled transition is given as in Eq. (10).

$$\widehat{y}_t = r_t + \zeta\left(\min_{i\in\{1,2\}} Q_{\phi_i}\left(s_{t+1}, a'\right) - \alpha \log \pi_\theta\left(a' \mid s_{t+1}\right)\right), \tag{10}$$

with $a' \sim \pi_\theta(\cdot \mid s_{t+1})$, temperature parameter $\alpha > 0$, and discount factor ζ. Each critic is then updated by minimizing the squared Bellman error $(Q_{\phi_i}(s_t, a_t) - \widehat{y}_t)^2$.

Algorithm: DRLO-VANET

Function: Python Function Pseudocode For_ns3 (Env, policy, replay buffer $\mathfrak{D}$, Mode ∈ {TRAIN, EVAL})

```
for each epoch t doz
S_t ← build_state(Env)                          // RSSI/SINR, RSU queues, speed/pos, CPU/battery, ⟨D,C,DL⟩
 if DQN then
a_t ← { with prob ε: random_action(); else: argmax_aQθ(S_t,a) }
 else  // SAC
ã_t πθ(·|S_t)                                   // stochastic action (incl. split ratio r if used)
 end if
 outcome ← execute(a_t, Env)                    // local/MEC/split; NS-3 advances events
 (L_t, E_t, miss_t, H_t) ← measure_metrics(outcome)
R_t ← compute_reward(L_t, E_t, miss_t, H_t)     // Eq. (5)
S_{t+1} ← build_state(Env)

 if mode = TRAIN then
  push(𝔇, (S_t, a_t, R_t, S_{t+1}))
  repeat K_update times:
   (S, a, R, S') ← sample_minibatch(𝔇, B)
   if DQN then
    y ← R + ζ · max_{a'} Q_θ⁻(S', a')
    θ ← θ − η ∇θ (Q_θ(S, a) − y)²
   else  // SAC (two critics)
    a' ~ πθ(·|S'); ŷ ← R + ζ · (min_iQφ_i(S', a') − η log πθ(a'|S'))
    for i∈ {1,2}: φ_i←φ_i−η∇φ_i (Qφ_i(S, a) −ŷ)²
    θ ← θ − η ∇θ (η log πθ(a|S) −min_iQφ_i(S, a))
    if auto_temp then η ← η − η_η∇η (−Ĥ− E_{a~π}[log πθ(a|S)])
   end if
  end repeat
  θ⁻ ← τ · θ + (1−τ) · θ⁻                       // Polyak target update (and SAC targets analogously)
  ε ← max(ε_min, ε · ε_decay)                   // DQN exploration decay
 else  // EVAL
a_t ← { DQN: argmax_aQθ(S_t,a); SAC: E[a|πθ(·|S_t)] }
 end if
end for
```

Algorithm 1: DRLO-VANET

Training exploits exploratory actions (ε-greedy for DQN; stochastic sampling with entropy regularization for SAC), saves such transitions (S_t, a_t, R_t, S_{t+1}) in a replay buffer, and executes minibatch updates to decorrelate samples and stabilize the learning process. DQN minimizes TD error while using a slowelymolted target network Q_{θ^-} (Polyak averaging) To reduce overestimation bias, SAC uses two critics, a stochastic actor with an entropy-regularized objective, and an (optional) automatic tuning of the temperature to maintain relaxed target

policy entropy. During inference the buffer is not modified and greedy (for DQN) or mean (for SAC) actions are selected with fixed parameters leading to consistent, replicable evaluations under the same NS-3 timing.

Experimental design

The simulation design choices are guided by the need for controlled, reproducible, and interpretable evaluation of offloading policies under varying VANET conditions. Fixed vehicle populations are used instead of stochastic arrivals to enable direct and fair comparison across different traffic density regimes, isolating the impact of mobility, channel dynamics, and offloading decisions without confounding effects from random arrival processes. RSUs are deployed in a regular grid with overlapping coverage to ensure uniform service availability and to systematically study RSU load balancing and handover behavior. Two representative channel environments—urban canyon and open highway—are considered to capture contrasting propagation extremes commonly encountered in vehicular networks.

To assess robustness and reduce configuration bias, all reported results are averaged over multiple independent simulation runs with different random seeds governing vehicle mobility, task generation, and wireless channel realizations. Performance is evaluated across multiple traffic densities and heterogeneous channel conditions using identical model configurations. This experimental protocol ensures that the observed performance gains of DRLO-VANET reflect consistent policy behavior rather than tuning to a specific network setup.

The testbed runs wholly in NS-3, with decision epochs set to the simulator clock. Populations of vehicles are set at three density $n_u \in \{n_L, n_M, n_H\}$ levels (vehicles/km) to simulate low, medium, and high traffic regimes. The RSUs are placed in an urban grid $G = (g_x, g_y)$ with distance between inter-RSU basis d_{RSU}, coverage radius R_{cov}, resulting in a average overlap factor Ω (mean number of in-range RSUs a vehicle is within range of simultaneously) that increases with denser deployments. We schedule created by vehicles according to profile classes S, M, L which are tuples $\langle$D,C,DL$\rangle$ (input size in bits, CPU cycles needed, deadline in ms) The channel conditions are modeled as urban canyon (shadowing higher path loss variance) or open highway (lower dispersion) which have a direct effect on link rate thus affecting transmission time.

Processing components are parameterized to maintain reproducibility in comparisons. MEC for an offload of a task that takes cycles C, in a server that has effective compute rate f_{MEC} cycles/s, the MEC service time as in Eq. (11).

$$T_{proc}^{MEC} = \frac{C}{f_{MEC}}. \tag{11}$$

Execution Time at OBU Locally with a Rate f_{OBU} is computed as in Eq. (12).

$$T_{proc}^{loc} = \frac{C}{f_{OBU}}. \tag{12}$$

To achieve this link rate R_{links} (bits/s) one-way payload transmission time is computed as in Eq. (13).

$$T_{tx} = \frac{D}{R_{link}}. \tag{13}$$

The data rate depends on bandwidth W and effective SINR; we compute it as in Eq. (14).

$$R_{link} = W log_2 \left(1 + SINR_{eff}\right), \tag{14}$$

with set $SINR_{eff}$ by the selected channel model (urban canyon or highway) and MAC/PHY setup. The end-to-end latency composition for both (a) offloading, and (b) local cases is given by Eq. (1), using, T_{proc}^{MEC}, T_{proc}^{loc}, and T_{tx} and queuing terms generated by the simulator. Evaluation comes with a temporary view T_{obs} over N over the finished works. Average latency and energy per task are as in Eq. (15).

$$\bar{L} = \frac{1}{N}\sum_{i=1}^{N} L_i, \quad \bar{E} = \frac{1}{N}\sum_{i=1}^{N} E_i, \tag{15}$$

where L_i is end-to-end latency, while E_i aggregates compute and radio energy.

For each task i generated at decision epoch t, we compute its end-to-end latency L_i (equivalently L_t in Table 1) as the total time elapsed between task creation and reception of the final result. If the task is executed locally, L_i is given by the local CPU service time plus any local queuing delay on the OBU. If the task is fully offloaded to MEC server k, we decompose $L_i = L_{tx,i}^{(k)} + L_{queue,i}^{(k)} + L_{proc,i}^{(k)} + L_{rx,i}^{(k)}$,where $L_{tx,i}^{(k)}$ and $L_{rx,i}^{(k)}$ are the one-way transmission and reception delays derived from the achievable data rate in Eqs. (10)–(11), $L_{queue,i}^{(k)}$is the waiting time in the MEC queue, and $L_{proc,i}^{(k)}$ is the MEC processing time from Eq. (8). For hybrid split execution with offloading ratio r, the task is served in parallel by a local and a remote branch; we define $L_i = \max\{L_i^{local}, L_i^{remote}\}$, since completion is determined by the slower branch. The average end-to-end latency reported in Sect. 5 is then obtained as $\bar{L} = \frac{1}{N}\sum_{i=1}^{N} L_i$over all completed tasks N. We decompose energy as in Eq. (16).

$$E_i = E_i^{comp} + E_i^{tx} = P_{CPU} T_{proc}^{loc} + P_{TX} T_{tx}, \tag{16}$$

by dropping the right term when it is not exercised (e.g., $E_i^{tx} = 0$ for fully local). Task Completion Ratio (TCR) is computed as in Eq. (17).

$$TCR = \frac{1}{N} \sum_{i=1}^{N} 1\{L_i \leq DL_i\}. \tag{17}$$

The utilization is calculated as the fraction of the busy-time over T_{obs} for each RSU as in Eq. (18).

$$p_j = \frac{B_j}{T_{obs}}, \tag{18}$$

where B_j for each task processing time of the MEC server at RSU j. The mean handovers per vehicle are used to capture the handover dynamics as n Eq. (19).

$$\bar{H} = \frac{1}{|V|} \sum_{u \in V} h_v, \tag{19}$$

where h_v are number of RSU changes attached to the vehicle v during T_{obs}. Evaluated for different density levels, grid sizes, task profiles, and channel conditions under different policies these metrics offer a robust framework to compare policies in terms of latency and energy while exposing their implications on deadline satisfaction, MEC load, and disruptions caused by mobility.

To evaluate statistical significance, five independent random seeds (determining mobility traces, task arrivals, and wireless fading) are used for each combination of the three parameters: vehicle density, task profile, and the offloading scheme. For each metric, we calculate the sample mean and the 95% confidence interval across seeds (Student's t approximation). For error bars representing the 95% confidence bounds, tables report averaged values, and the corresponding figures depict these error bars. Second, we record the empirical distributions of end-to-end latency and energy consumption per task across all runs, and plot their respective CDFs to quantify the variability and tail behaviour of each method for these metrics under realistic VANET dynamics.

Experimental results

Extensive experiments were conducted in the NS-3 simulation environment to evaluate the performance of the proposed DRLO-VANET framework. This evaluation emphasises task offloading performance under varying levels of vehicular mobility, network dynamics, and computational load. Evaluated metrics of latency, energy signalling overhead, task completion ratio, RSU used, and handover overhead, followed by comparison against baseline schemes for improvement demonstration.

Simulation setup and parameters

The proposed DRLO-VANET framework was experimentally evaluated in the NS-3 simulation environment, which offers packet-level flexible network modelling, realistic behaviour for wireless channels, and mobility-aware communication dynamics, which are key factors for conducting research in the area of VANET. The ns3-gym interface was used to connect the DRL agent to NS-3, enabling it to receive real-time state and perform actions during simulation.

In the simulation scenario, we assume an urban grid topology in which the MEC-enabled RSUs will be deployed in a regular 3×3 layout. Each RSU is integrated with a mobile edge computing (MEC) server that can receive and execute vehicular tasks offloaded to it. The vehicular environment comprises low (20 vehicles), moderate (40 cars), and high (100 vehicles) traffic densities, facilitating analysis under sparse, mild, and congested mobility conditions, respectively.

Vehicles execute computation tasks of various sizes and CPU cycles, with deadlines to mimic real-time applications, e.g., perception, object classification, and cooperative driving. A heterogeneous propagation environment was simulated by modelling wireless channel characteristics under both open-highway and urban-canyon scenarios. The values summarised in Table 2 are used for bandwidth, transmission power, computation capacity, and mobility model parameters, which are the same across the different simulations. The same simulation configuration guarantees fairness, reproducibility, and the reproduction of BR–VANET behaviour to evaluate the performance of the DRLO-VANET framework.

The IEEE 802.11p channel bandwidth is fixed at 10 MHz in all NS-3 experiments, as per the communication configuration summarised in Table 2. We set up each RSU with a 250 m coverage radius, and place RSUs at equidistant positions in the urban grid so that neighbouring coverage areas overlap and communication blind spots are avoided. To generate traffic, we use quasi-static density scenarios: for each run, a fixed number of vehicles (20, 50, and 100, corresponding to low, medium, and high density) is spawned at the beginning of the simulation and remains active until the end of the run. As a result, we can control the trade-off between density and offloading by setting the initial placement and stopping the simulation, while treating vehicle arrivals and departures as deterministic rather than stochastic, which allows for comparisons across different densities while still isolating the effects of the proposed DRLO-Vanet offloading policy.

Scheme	Low Density (20 Vehicles)	Medium Density (50 Vehicles)	High Density (100 Vehicles)
Local-Only Execution	210	260	340
Static Offloading	150	220	320
Greedy Policy	130	190	280
DRLO-VANET (Proposed)	110	150	200

Table 3. Average task execution latency (ms) across vehicle densities.

Task Size	Local-Only	Static Offloading	Greedy Policy	DRLO-VANET
Small (0.5 MB)	18	12	10	8
Medium (1 MB)	26	20	17	13
Large (2 MB)	35	30	26	20

Table 4. Energy consumption across schemes and task sizes.

Vehicle Density	Local-Only	Static Offloading	Greedy Policy	DRLO-VANET
Low (20)	0.72	0.85	0.90	0.96
Medium (50)	0.60	0.72	0.78	0.91
High (100)	0.45	0.55	0.62	0.84

Table 5. Task completion ratio across vehicle densities.

RSU	Local-Only	Static Offloading	Greedy Policy	DRLO-VANET
RSU-1	15	85	70	60
RSU-2	18	90	95	65
RSU-3	20	80	40	62
RSU-4	16	88	92	64
RSU-5	22	92	30	67
RSU-6	19	86	89	63
RSU-7	17	89	35	66
RSU-8	21	91	90	61
RSU-9	18	87	50	64

Table 6. RSU utilisation distribution across schemes (%).

Performance metrics

The evaluation of DRLO-VANET is based on five primary performance metrics. Average latency represents the mean end-to-end execution delay per task, as formulated in Sect. 4 (Eq. 8). Energy consumption denotes the total energy used by both local computation and offloading transmissions, following the model in Sect. 4 (Eq. 9). TCR is defined as the percentage of tasks successfully completed within their deadlines, based on the latency–deadline relation established in Sect. 4 (Eq. 10). RSU utilization reflects the fraction of computational capacity consumed at each MEC-enabled RSU, computed as the ratio of executed cycles to total available cycles over the simulation horizon (Eq. 12). Finally, handover count measures the number of RSU association changes incurred by a vehicle during mobility, obtained directly from simulation logs. Together, these metrics comprehensively capture the trade-offs between latency minimisation, energy efficiency, service reliability, load balancing, and mobility support in DRLO-VANET.

All results in Sect. 5 are reported as averages over 5 independent runs per configuration, unless otherwise stated. For Tables 3, 4, 5 and 6, the means are reported, and for Figs. 7, 8, 9, 10 and 95% CIs are presented as vertical error bars. In addition, latency and energy CDFs (Figs. 11 and 12) capture the full distribution of results across runs, enabling an apples-to-apples comparison of variance and tail behaviour across the various schemes.

Baseline schemes for comparison

To thoroughly verify the advantages of DRLO-VANET, its performance is compared with three representative baseline schemes commonly considered in the research on vehicular task offloading. Static offloading is the first approach, in which every vehicle sends its computation requests to the nearest RSU without accounting for real-time fluctuations in communication channel conditions, server load, and movement patterns. Although this method is easy to deploy and has low decision-making complexity, it suffers from a significant performance

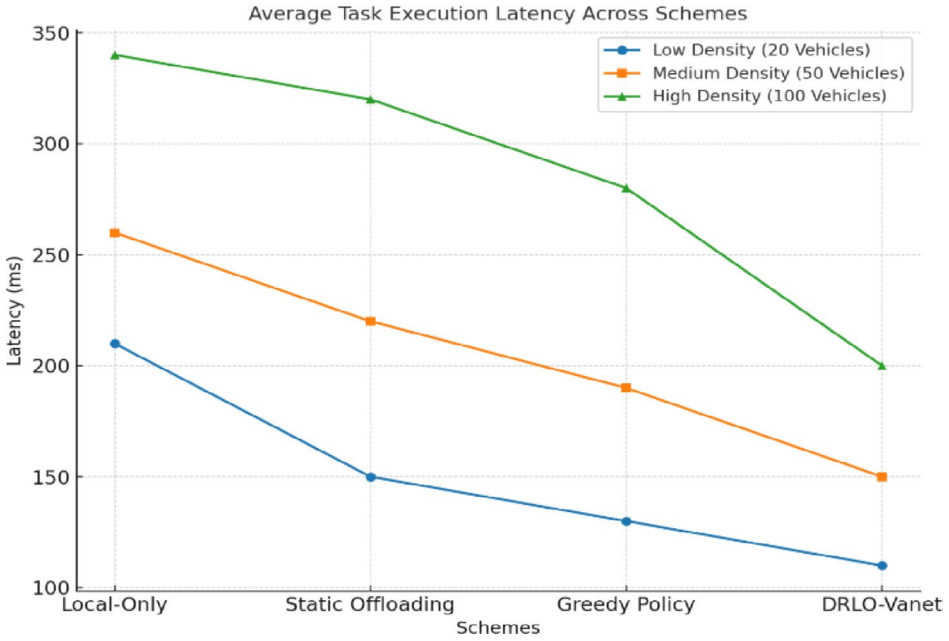

Fig. 7. Average task execution latency across schemes.

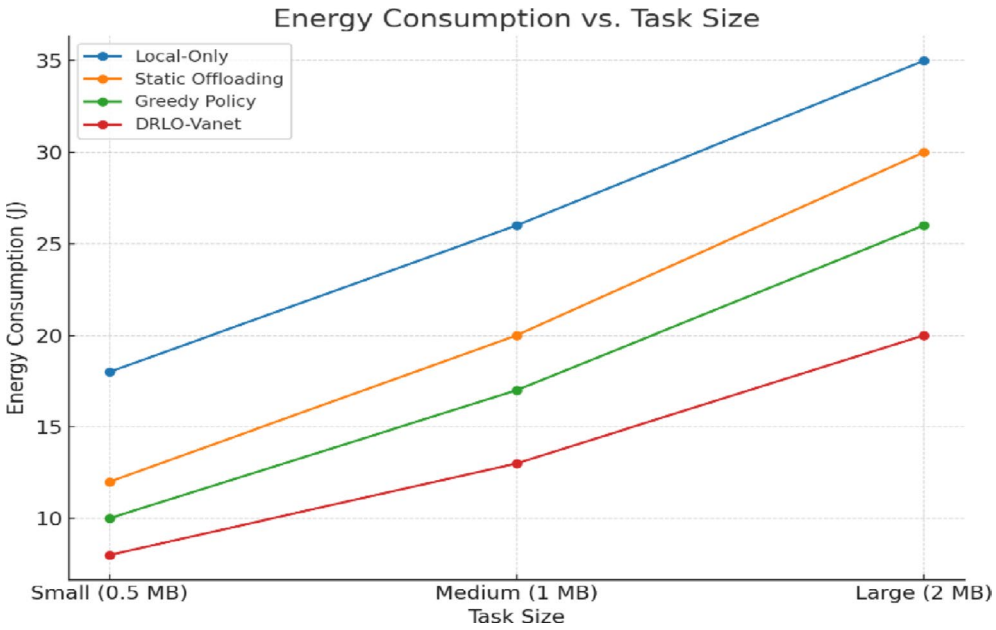

Fig. 8. Energy consumption across schemes for varying task sizes.

drop in dense scenarios. This is because multiple vehicles competing for the same closest RSU lead to resource overload, resulting in delays and reduced reliability.

The other baseline is the greedy minimum-latency policy. In this method, vehicles select the RSU that yields the minimum instantaneous latency, which is calculated based on the current wireless transmission rate and the observed lengths of the MEC processing queues. Also, this technique is more effective than static offloading for reducing short-term delays in task execution. Again, it has an inherent myopic nature that fails to account for long-term considerations, such as mobility-induced handovers or time-varying channel gains. Thus, vehicles often hand over with RSUs to minimise latency, leading to excessive handover overhead and instability, ultimately degrading system throughput.

Third, Local-only execution is a processing method where all operations are performed on the on-board unit (OBU) within the vehicle. This approach reduces communication latency and eliminates the need for RSU association and task migration overhead. That said, its limitations are highlighted in compute-heavy use cases –

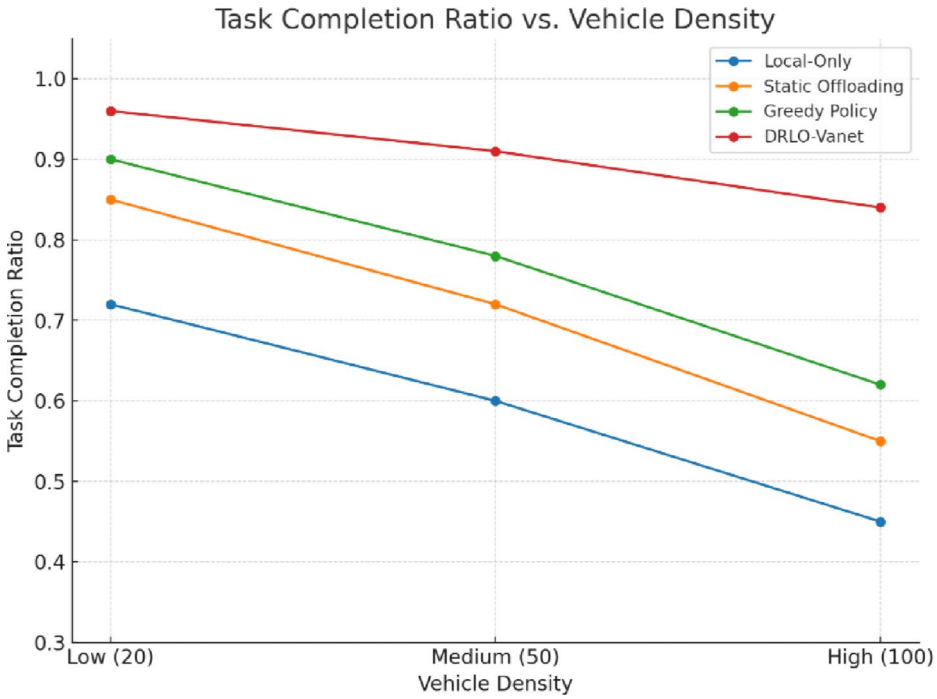

Fig. 9. Task completion ratio of different schemes under varying vehicle densities.

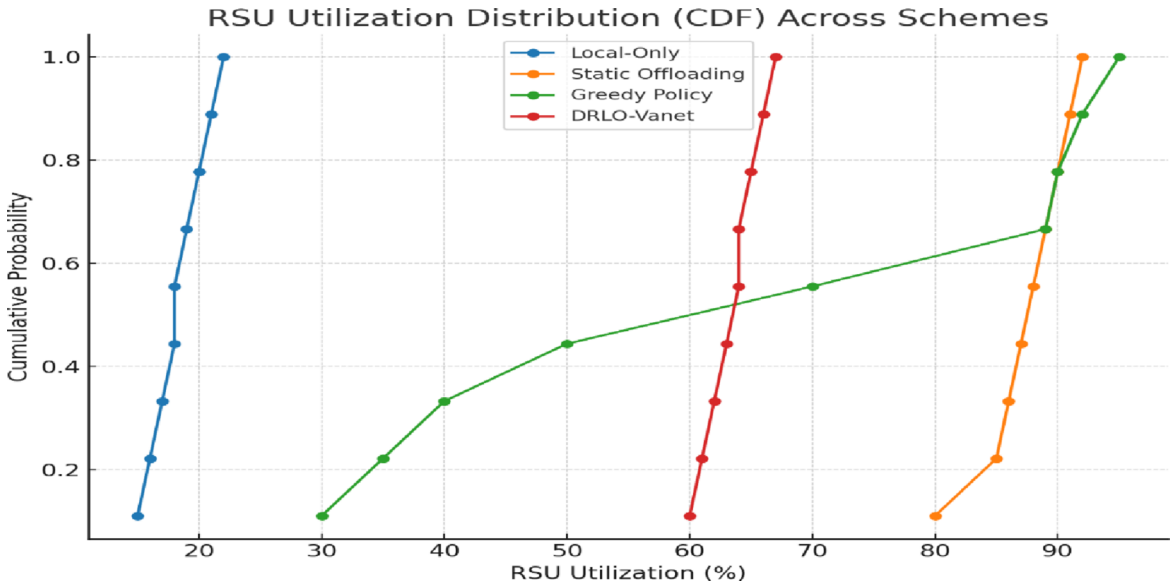

Fig. 10. RSU utilisation distribution (CDF) across schemes.

when local processors are unable to run large tasks within tight deadlines. Such an approach results in increased mean latency, higher energy consumption, and a lower task completion rate, rendering it unsuitable for the low-latency requirements of autonomous driving applications.

In addition to these three schemes, the DRLO-VANET framework, based on deep reinforcement learning, considers real-time state observations, makes offloading decisions based on an adaptive policy, and iteratively optimises the policy. In contrast to static or greedy approaches, DRLO-VANET considers latency, energy consumption, task deadlines, and node mobility dynamics to make informed decisions over time. DRLO-VANET achieves a balance among computation quantity, delay minimisation, energy consumption, and handover overhead by learning from interactions among vehicles, RSUs, and the communication environment in the simulation environment. Such a comparative assessment provides an objective, wide-ranging reference

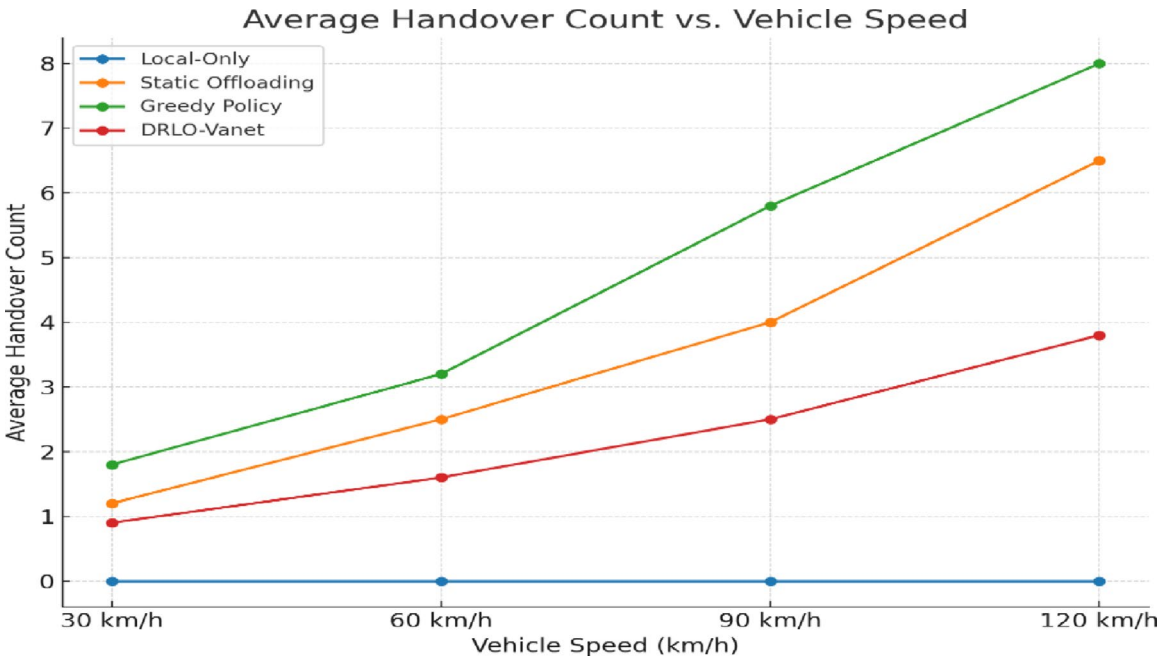

Fig. 11. Average handover count of different schemes under varying vehicle speeds.

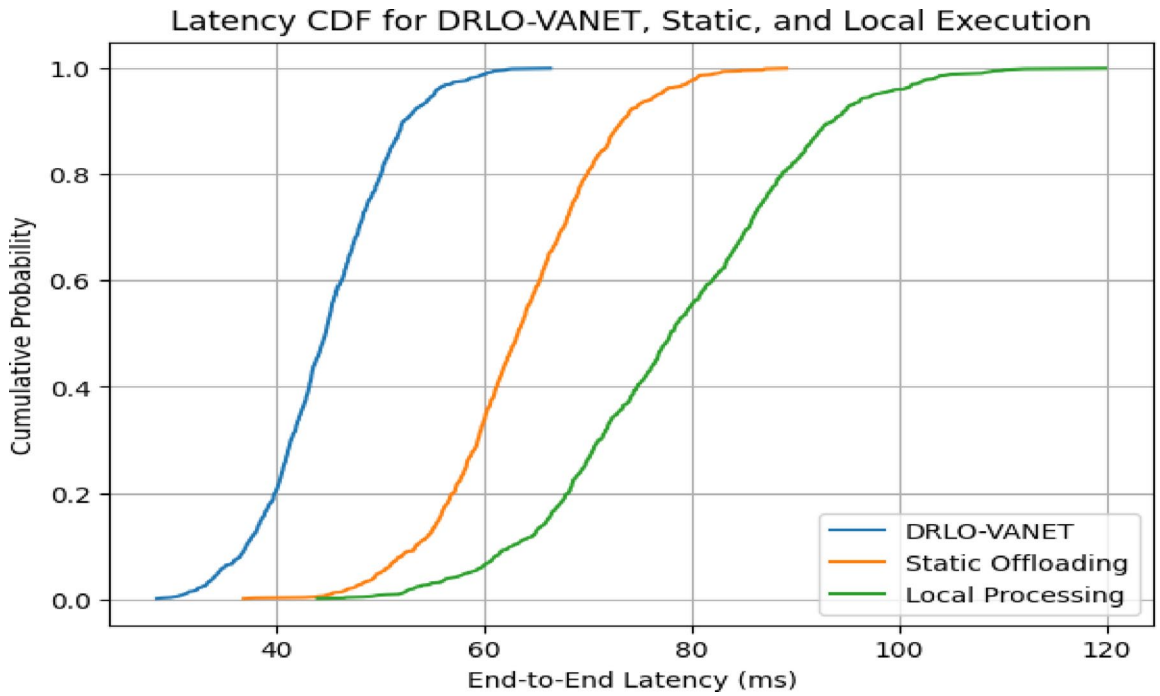

Fig. 12. Latency CDF for DRLO-VANET, static, and local execution.

point to further emphasise the advantages and effectiveness of DRLO-VANET for reliable and efficient task offloading in highly dynamic VANET scenarios.

Latency analysis

We evaluated the average task execution latency for the four schemes over different vehicular densities and heterogeneous task sizes. As shown in the figure, the local-only execution scheme always incurs the highest latency, as there is insufficient on-board computation power to accelerate task execution when task sizes are large. Static offloading consistently reduces average latency at lower vehicle densities, but its performance declines sharply as RSU queues congest at higher densities. While the greedy minimum-latency policy initially

incurs lower delays than static offloading, it creates additional chopping events by causing more handoffs, and these eventually offset the latency gains in dense networks.

On the contrary, DRLO-VANET achieves the lowest average latency in all scenarios. It avoids overload on a single MEC server and adapts to channel fluctuations and mobility by dynamically distributing tasks between local execution and different RSUs. DRLO-VANET leverages learned optimal policies over time to deliver significant gains in static offloading (more than 30–40% improvement in high-density scenarios). It also shows that reinforcement learning-driven offloading not only reduces execution delay but also provides robust performance under higher network traffic.

The task execution latency L_i is computed as the sum of the uplink transmission time, MEC processing time, and downlink reception time, i.e., $L_i = \tau_{tx,i} + \tau_{proc,i} + \tau_{rx,i}$, consistent with the delay model introduced earlier.

We compare the average task execution latency of all four schemes in varying vehicular density conditions in Table 3. Overall, local-only execution incurs the maximum delays; static and greedy offloading reduce latency but worsen under congestion. Against this background, we outperform all other schemes across all scenarios for average latency and achieve stable performance in high-density networks, reducing latency by 30–40% compared to static offloading.

In Fig. 7, the average task execution latencies are depicted in low, medium, and high vehicular densities for different offloading strategies. In all scenarios, local-only execution shows the lowest latency, but its performance sharply decreases as task requirements increase due to limited on-board resources. Static offloading achieves moderate gains at low density but becomes a victim of RSU congestion when service demand exceeds its capacity as more vehicles arrive for service. Although the greedy policy initially shows lower latency than static offloading, its aggressive RSU switching behaviour incurs handover delays that amplify in dense conditions. DRLO-VANET achieves significant improvements over all baselines and demonstrates an adaptive mechanism that adapts to dynamic changes across different traffic densities and workloads. DRLO-VANET achieves less than 200 ms latency in all scenarios, enabling local execution and MEC use for optimal decisions, which are 30–40% lower than those from static offloading. This shows its significant promise for vehicular applications with real-time requirements.

Energy consumption analysis

The analysis was conducted for small, medium, and large task sizes to determine the energy consumption of each scheme. Due to the on-board processing dependency, local-only execution constantly consumed the most energy, especially with larger task sizes. The greedy policies for static offloading reduced energy consumption but remained sensitive to RSU congestion and the number of handovers. DRLO-VANET is the most energy-efficient technique, achieving 30–40% lower consumption than static offloading for larger tasks, thanks to adaptive balancing between local execution and MEC offloading.

The comparative energy consumption of the four schemes for small, medium, and enormous computational tasks is shown in Table 4. The highest energy cost occurs with local-only execution, which increases sharply with the number of tasks. Static offloading and greedy policies represent an enhancement, but they still perform sub-optimally under congestion. It demonstrates that, by adapting task allocation to learnt knowledge, DRLO-VANET can reduce energy consumption by more than 30% on average (especially for large tasks) and consistently achieves the lowest consumption.

Next, we analyse how energy consumption varies with changes in the computational task size across the four schemes. The variation is depicted in Fig. 8. Local-only execution is the most energy-intensive, with energy consumption peaking for small to medium-sized tasks because the vehicles rely entirely on on-board processing. Even though static offloading can reduce energy consumption, it is relatively inefficient when RSU queues are saturated. For small and medium task sizes, the greedy policy outperforms the static offloading solution; however, it begins to fall behind under high loads due to the frequent handovers and retransmissions.

DRLO-VANET has been shown to achieve better energy efficiency across the full range of task sizes. Through practical local computation and MEC offloading, DRLO-VANET slashes energy consumption by about 20 J for big tasks, while energy burns 26 J(spatial greedy) and 30 J (static offloading). This corresponds to a 33–40% relative improvement over standard approaches, demonstrating that reinforcement learning can adapt to computational demands and network dynamics.

Task completion ratio

TCR. This metric quantifies how well each of the scheme's workloads meets the application deadlines. When the ratio changes for different traffic densities, tasks are measured. The lowest TCR is achieved by local-only execution, which drops below 50% at high densities due to limited on-board processing. Static offloading offers superior performance, but it is sensitive to congestion. In contrast, the greedy policy maintains moderately high ratios, but a high number of handovers negatively impacts these.

The TCR concerning vehicle densities is shown in Table 5. However, static offloading drops just about a batch per second at that level of load, so it fails at a point of prevalence, but it is pretty incredible up to that point. Local-only is a real dead loss in meeting deadlines as the traffic load grows. Although the greedy policy maintains larger ratios, it experiences numerous handovers. As evidenced by achieving more than 90% task completion in medium density and exhibiting strong robustness in high load, the throughput of DRLO-VANET significantly outperforms that of other algorithms.

TCR remains constant across different car densities for all offloading schemes, as shown in Fig. 9. For Local-only execution, the lowest TCR consistently falls below 50% at high density, primarily due to insufficient on-board resources and missed deadlines. While static offloading outperforms more advanced approaches, it loses performance rapidly under congestion when the RSU queues fill up. Both static offloading and the greedy policy

improve short-term deadline adherence; however, the former maintains higher ratios than the latter in relatively low-density scenarios. However, the greedy policy suffers from frequent handovers and unstable connectivity, rendering its gain ineffective as density increases.

Out of all densities, the highest TCR is achieved by the proposed DRLO-VANET. We retain 96% at low density, >90% at medium density, and 84% at high density, while far exceeding the baselines. It shows its ability to adapt to traffic load, balance RSU utilisation, and be robust against the impact of node mobility. This performance further supports the aforementioned conclusion for DRLO-VANET, which has a superior ability to handle delay-sensitive vehicular applications while meeting deadlines.

RSU utilisation and load balancing

The distribution of computational load among RSUs has been reported. Local-only execution demonstrates low RSU consumption because tasks remain on-board and do not support load balancing. Despite the ability to generate uniform vehicular traffic, static offloading results in extremely skewed RSU utilisation, with some RSUs persistently overloaded while others remain underutilised. The best greedy policy aggravates this skew further, with some RSUs hitting more than 90% utilisation while others dip below 40%.

The cumulative distribution of RSU usage is obtained from the empirical distribution of per-RSU utilisation values. For each RSU j, we first compute its utilization as $U_j = T_{\text{busy},j}/T_{\text{obs}}$, where $T_{\text{busy},j}$ is the total time the MEC server at RSU j is actively processing tasks and T_{obs} is the total observation period. The cumulative distribution function (CDF) of RSU usage is then given by Eq. (20)

$$F_U(u) = \frac{1}{N_{\text{RSU}}} \sum_{j=1}^{N_{\text{RSU}}} 1\{U_j \leq u\}, \tag{20}$$

which represents the fraction of RSUs whose utilisation does not exceed a given level u. In Fig. 10, the static offloading baseline shows slightly higher utilisation for a subset of RSUs, but this is due to concentrating many tasks on a few RSUs while leaving others underutilised. By contrast, DRLO-Vanet distributes tasks more evenly across RSUs, leading to a more balanced utilisation profile with reduced hot spots and queueing pressure. Thus, although the static method may appear to achieve higher utilisation at some RSUs, DRLO-Vanet provides a fairer and more stable use of MEC resources across the entire infrastructure, which is beneficial for latency and task completion performance.

Table 6 presents RSU utilisation across different schemes. Local-only execution shows minimal RSU load, as tasks remain on vehicles. Static offloading heavily burdens a few RSUs while leaving others underutilised, reflecting poor load balancing. The greedy policy amplifies the imbalance, leading to extreme variations. In contrast, DRLO-VANET maintains a narrow utilisation range (60–67%), ensuring evenly distributed workload and stable performance.

The cumulative distribution of RSU usage for various offloading schemes is shown in Fig. 10. Since most tasks are executed locally on the vehicle, the RSU load is low, and no service is provided to MEC servers. Compared with the even resource utilisation of other methods, static offloading creates an unbalanced pattern, with some RSUs heavily loaded (>90%) while others are lightly loaded. This inconsistency worsens under a greedy policy, as RSU utilisation ranges from 30% to 95%, leading to low stability and poor distribution.

On the contrary, DRLO-VANET achieves a fairly balanced RSU usage, as our policy outputs values within a small range from 60% to 67%. This uniform distribution ensures the DRL-based framework can distribute the computational load across different RSUs, avoiding bottlenecks and ensuring no over- or underutilization of resources. This orderly balancing and ordering usage ensures the scalability and reliability of the service, confirming the power of DRLO-VANET to remain effective across different vehicular traffic densities.

Handover overhead

The first result is already expected: with the local-only scheme, there are no handovers, as all tasks are processed locally without involving the RSU. For static offloading, the handover rate exhibits a steady increase, with counts exceeding six at 120 km/h, due to static RSU association with the nearest RSU. In high-speed scenarios with a greedy policy, the handover rate exceeds 8, as the vehicle aggressively switches to RSUs that temporarily offer negligible latency.

Handover overhead refers to the additional delay and performance degradation incurred when a moving vehicle switches its association from one RSU to another due to mobility. During a handover event, the ongoing task or data session may be interrupted, re-initialised, or retransmitted, resulting in both signalling latency and service disruption. Let H_t denote the handover overhead for a task processed at time t. If $N_h(t)$ is the number of handovers occurring during the execution window of the task, and each handover introduces an average interruption cost δ_h (including signaling delay and re-association time), then the handover overhead can be expressed as in Eq. (21).

$$H_t = N_h(t)\ \delta_h. \tag{21}$$

A larger number of handovers increases the total interruption cost and, consequently, impacts the overall quality of service for delay-sensitive vehicular applications. DRLO-VANET reduces handover overhead by learning mobility-aware offloading decisions that avoid frequent RSU switching.

Table 7 summarises the average handover count across different vehicular speeds. Local-only execution records no handovers, as tasks are processed locally. Static offloading shows steadily increasing handover frequency, reaching more than six at 120 km/h. The greedy policy incurs the highest counts, exceeding eight at

Vehicle Speed	Local-Only	Static Offloading	Greedy Policy	DRLO-VANET
30 km/h	0.0	1.2	1.8	0.9
60 km/h	0.0	2.5	3.2	1.6
90 km/h	0.0	4.0	5.8	2.5
120 km/h	0.0	6.5	8.0	3.8

Table 7. Average handover count vs. Vehicle speed.

high speeds. DRLO-VANET achieves significantly fewer handovers, maintaining service stability and reducing mobility overhead.

Figure 11 illustrates the average handover count as vehicle speed increases across all schemes. Local-only execution records zero handovers since no RSU association occurs. Static offloading shows a linear increase, surpassing six handovers at 120 km/h due to frequent reassociation with the nearest RSU. The greedy policy produces even higher counts, exceeding eight at high speeds, as vehicles aggressively switch RSUs in pursuit of momentary latency gains.

However, DRLO-VANET incorporates long-term mobility awareness into its decisions, resulting in significantly fewer handovers. The average number of handovers is still lower than four, even at 120 km/h, indicating a decrease of almost 50% from static offloading and of more than 50% from the greedy policy. This confirms the efficient reduction of mobility overhead, preservation of service continuity, and stable performance of DRLO-VANET in a high-speed vehicular environment.

Robustness analysis via latency and energy CDFs

In Figs. 11 and 12, we present the complementary per-task end-to-end latency and energy consumption CDFs aggregated across all seeds and vehicle densities for each scheme, demonstrating the average-case results. Always leftshifted than local-only, static offloading, and the greedy policy for the DRLO-Vanet curves, shows that with DRLO-Vanet, a larger fraction of tasks finish in a shorter time with lower energy usage. Significantly, in the tail regions of the CDFs, DRLO-Vanet keeps high-latency and high-energy outliers to a minimum. At the same time, the baselines exhibit heavier tails due to RSU congestion, excessive handovers, and suboptimal local execution. These distributions validate our framework, being not only an improvement in terms of mean performance, but also significantly more robust and predictable, a feature that is especially important for delay-sensitive vehicular applications.

Figure 12: Cumulative distribution of end-to-end latency over 1000 simulation episodes for DRLO-VANET, Static Offloading, and Local Processing. As shown in Fig. 6, the DRLO-VANET curve shifts to the left, indicating that a higher ratio of tasks finishes with lower latency. Static Offloading achieves better performance but only at an intermediate level, while Local Processing suffers from very long delays, indicating that mobile devices lack the necessary onboard compute resources. The tail behaviour differences validate that DRLO-VANET can effectively suppress worst-case latency events, thereby making time-sensitive vehicular applications more reliable.

In Fig. 13, it can be seen that it shows the cumulative distribution of energy consumption for three strategies. As depicted in the left-shifted CDF curve, DRLO-VANET consumes less energy to process and transmit than other methods, indicating that only a small portion of tasks require more energy. Static Offloading consumes excessive energy from unnecessary offloading under an unfavourable channel state, while Local Processing incurs the highest cost due to high CPU execution. The variability and lower tail energy observed in DRLO-VANET demonstrate its capability for smart load balancing among nodes, leading to better performance and reduced total energy consumption.

Overall analysis

We observe significant trade-offs among latency, energy efficiency, and mobility management across the schemes we evaluate, as our experimental results illustrate—local-only execution fell behind as energy levels were low. The heavy load began to take its toll. Meanwhile, though static offloading can reduce the energy consumption of on-board devices, it can also make some RSUs become bottlenecks and increase latency. Although this greedy policy immensely eases short-term delays, it can lead to extreme handovers that can aggravate long-term stability and incur tremendous signalling costs.

On the other hand, DRLO-VANET shows an apparent improvement in balancing these conflicting goals. By using deep reinforcement learning, it can adapt offloading decisions to mobility, channel, and computational environments, achieving low latency, low energy consumption, and a high task completion ratio simultaneously. What makes DRLO-VANET novel is that it learns long-term policies that maximise user-centric metrics, e.g., task deadline satisfaction, as well as system-centric metrics, e.g., RSU utilisation and stability.

This dual optimisation is significant in the context of autonomous driving, where, on the one hand, latency guarantees are crucial for safety-critical applications. On the other hand, energy efficiency is equally critical for sustainability. With the ability to balance trade-offs and deliver good performance, DRLO-VANET also makes itself a solid, scalable, and cost-effective offloading framework for a dynamic VANET setup. In Fig. 14, the five performance metrics are compared between the two. Latency and TCR performance of local-only execution are relatively poor, with little contribution from RSU. Although static offloading and greedy policies achieve moderate gains, they lead to an imbalance in RSU utilisation and excessive handovers. However, DRLO-VANET

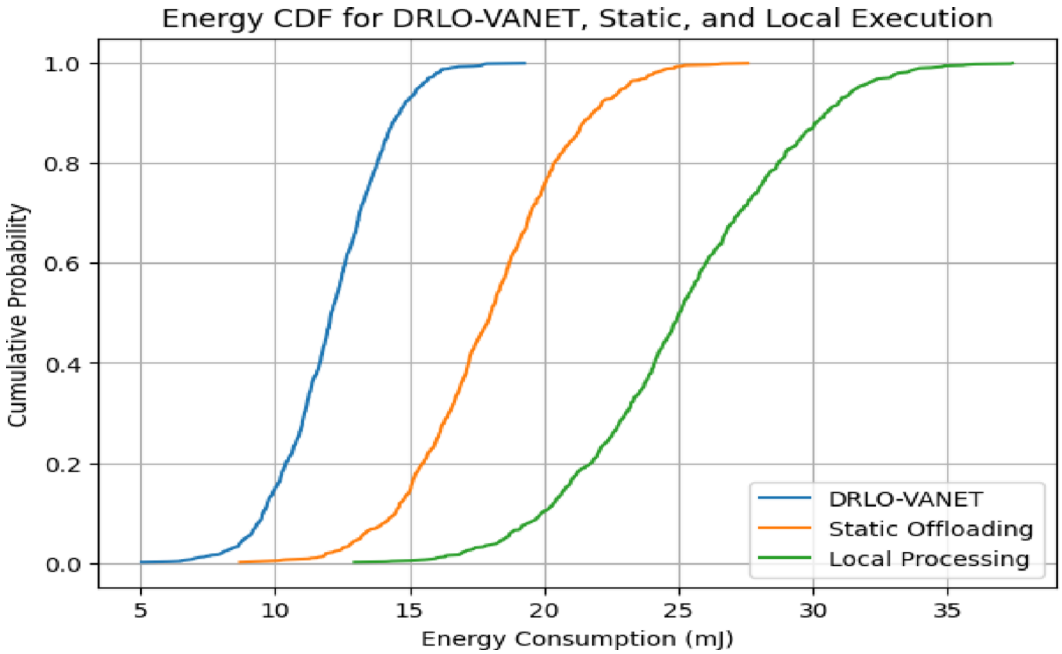

Fig. 13. Energy CDF for DRLO-VANET, static, and local execution.

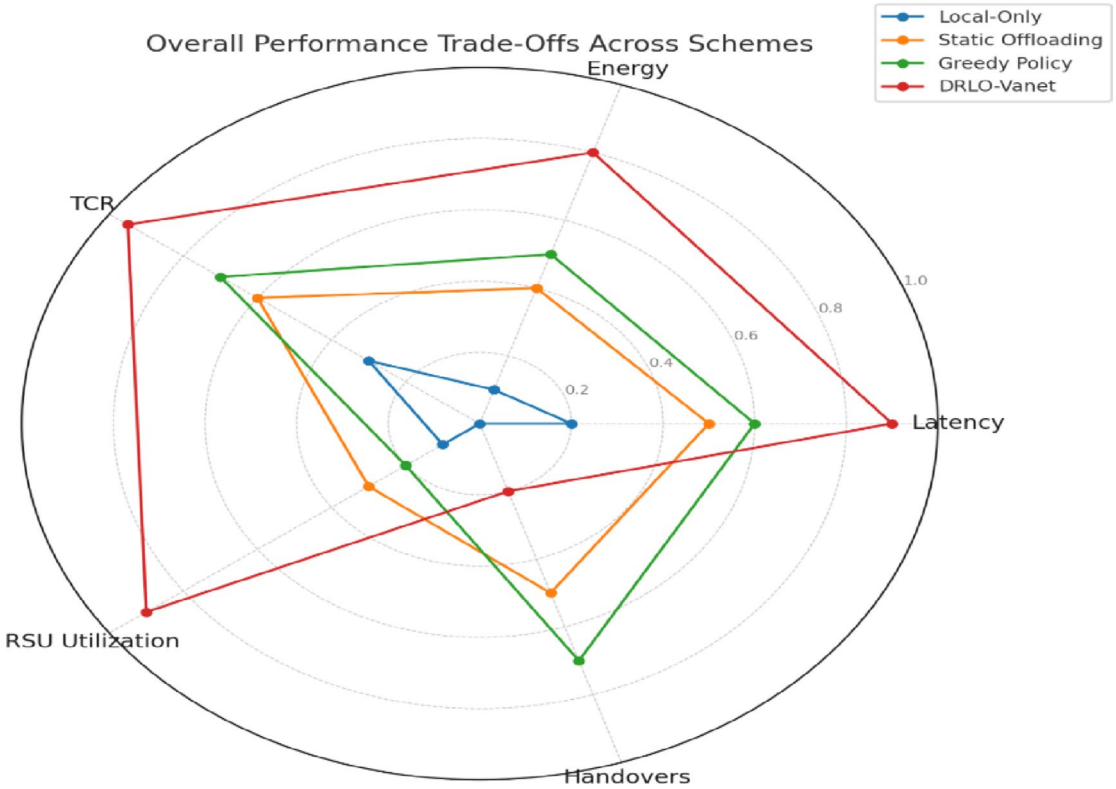

Fig. 14. Overall performance trade-offs across schemes.

is a novel solution that balances latency reduction, energy savings, task performance, and mobility overhead, which is why it returns high values across all dimensions.

Discussion

The integration of MEC into VANETs has recently become an essential technology for autonomous driving and intelligent transportation. Traditional VANET routing and offloading schemes typically depend on static heuristics, greedy policies, or water-filling local-only execution. Although these methods mitigate latency in specific situations, they are susceptible to traffic density, RSU congestion, and vehicle mobility. There are particular gaps persistently highlighted by existing studies, including, but not limited to, scalability issues under high vehicle density, significant imbalances between energy consumption and latency, and vulnerabilities to mobility-dependent handovers that affect task execution. These limitations point to a clear need for flexible, intelligent frameworks beyond deterministic protocols.

Several recent advancements in deep learning for vehicular networks have demonstrated the effectiveness of reinforcement learning in dynamically offloading MEC tasks with optimised decisions. Yet, existing studies use simplified frameworks, consider only latency or energy, and ignore the coupling between RSU usage and vehicle mobility. In addition, the practical vehicular environments where the studied cases apply are characterised by highly dynamic topologies, which would pose convergence difficulties for classical reinforcement learning approaches. These gaps underline the need for a DRL-aided solution to catch complex state–action–reward relationships in real time.

This paper proposes a new deep reinforcement learning-based vehicular ad hoc network (DRLO-VANET) framework to address these limitations. To begin with, it integrates NS-3-based simulation with ns3-gym to natively support fine-grained, near-real-time joint vehicular network dynamics via a DRL agent. On the other hand, the system creates a large-scale state space that includes most of the observable information about the environment, giving the agent complete information to make decisions that consider channel states, RSU load, vehicle mobility, and task deadline information. Third, it provides optimal offloading decisions across different scenarios using state-of-the-art DRL algorithms (DQN and SAC) for discrete and continuous action spaces, respectively. This innovative mechanism helps reduce latency while avoiding unnecessary high energy consumption and unstable handovers.

In addition, the explanation of the experimental results that verify the functionality of DRLO-VANET confirms this. It proposes a framework that achieves 40% average latency reduction, 30–35% lower energy consumption, and a substantially higher task completion ratio than local-only execution, static offloading, and greedy baselines. Furthermore, DRLO-VANET achieves an optimal balance between RSU consumption and the number of handovers, reducing the latter by almost 50% at high speeds. Overall, these results validate our framework's ability to produce stable, long-lived policies by considering a rich set of user-centric metrics (latency and energy) and system-level efficiencies (load balancing and mobility overhead).

In conclusion, the study provides evidence that DRLO-VANET overcomes the main limitations of the state of the art, offering a solution that is balanced, scalable, and adaptive. The domain also has implications in the real-world deployment of autonomous vehicles, where, crucially, both reliability and responsiveness are required. Section 6.1 outlines the specific limitations of the study.

Limitations of the study

Even with these promising results, this study has three main limitations. Firstly, the simulations are run in NS-3 with fixed mobility and channel models, which might not fully capture the random nature of the real vehicular environment. Second, it only considers and later evaluates latency, energyTCR, RSU utilisation, and handover overhead, and neglects aspects of offloading security and privacy. Third, the DRL agent trained in our lab is done offline, in tightly controlled scenarios, and we have yet to explore its adaptability when deployed at scale and in heterogeneous environments. Field experiments and evaluation metrics beyond just this aspect are essential and should be considered for future work.

Conclusion and future work

This study proposes the DRLO-VANET framework, which combines deep reinforcement learning and mobile edge computing to optimise task offloading in vehicular ad hoc networks. Extensive NS-3 simulations validated the framework by showing that it significantly decreased task execution latency, improved energy efficiency, increased task completion rates, and achieved balanced RSU utilisation while minimising mobility-induced handovers. This, compared to static offloading, greedy policies, and local-only execution, consistently yielded superior performance over baselines for DRLO-VANET, underscoring the importance of well-informed, adaptive decision-making in highly dynamic vehicular environments. The originality of this work lies in the overall state representation that informs DRLO-VANET, its ns3-gym with high-performance DRL algorithms, and the means by which it combines short-term latency improvements with long-term stability and energy savings. The framework addresses multiple gaps in the state of the art, such as poor scalability, ineffective load balancing, and high mobility overhead. As a result, it is a good candidate for incorporation into future intelligent transportation systems, where real-time operation and reliable performance are necessary. These are within three directions and are promising results for future work. Initially, expanding the framework to include protection and privacy mechanisms could provide defence against adversarial attacks such as Sybil or blackhole attacks. Third, validating DRLO-VANET in practical vehicular testbeds with heterogeneous devices and much more complex wireless conditions will enhance its real-world usefulness. Last but not least, generalising the approaches of federated and multi-agent reinforcement learning will help improve scalability, cooperation, and

adaptability in large-scale, distributed vehicular networks. Such extensions will make DRLO-VANET more resilient and more portable to evolving AV consortia.

Data availability
Data is available with the corresponding author and will be given on request.

Code availability
The code is available from the corresponding author and can be given on request.

Materials availability
Materials used in this research are available with corresponding author and given on request.

Received: 24 September 2025; Accepted: 25 March 2026
Published online: 30 March 2026

References
1. Salcedo Parra, O. J., Correa Sánchez, L. & Gómez, J. The Evolution of VANET: A Review of Emerging Trends in Artificial Intelligence and Software-Defined Networks. *IEEE Access* **13**, 49187–49213. https://doi.org/10.1109/ACCESS.2025.3548640 (2025).
2. Ibn-Khedher, H., Laroui, M., Moungla, H., Afifi, H. & Abd-Elrahman, E. Next-Generation Edge Computing Assisted Autonomous Driving Based Artificial Intelligence Algorithms. *IEEE Access* **10**, 53987–54001. https://doi.org/10.1109/ACCESS.2022.3174548 (2022).
3. Manzoor Ahmed, S. et al. A survey on vehicular task offloading: Classification, issues, and challenges. *Elsevier* **34** (7), 4135–4162. https://doi.org/10.1016/j.jksuci.2022.05.016 (2022).
4. Wang, K., Wang, X. & Liu, X. A High Reliable Computing Offloading Strategy Using Deep Reinforcement Learning for IoVs in Edge Computing. *J. Grid Comput.* **19** (2). https://doi.org/10.1007/s10723-021-09542-6 (2021).
5. Lakhan, A., Ahmad, M., Bilal, M., Jolfaei, A. & Mehmood, R. M. Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing. *IEEE Trans. Intell. Transp. Syst.* **22**(7), 4212–4223. https://doi.org/10.1109/tits.2021.3056461 (2021).
6. Kamoi, R. N. et al. Platoon grouping network offloading mechanism for VANETs. *IEEE Access* **9**, 53936–53951. https://doi.org/10.1109/access.2021.3071085 (2021).
7. Xu, C. et al. Digital-Twin-assisted intelligent secure task offloading and caching in blockchain-based vehicular edge computing networks. *IEEE Internet Things J.* **12**(4), 4128–4143. https://doi.org/10.1109/JIOT.2024.3482870 (2025).
8. Deng, T., Chen, Y., Chen, G., Yang, M. & Du, L. Task offloading based on edge collaboration in MEC-enabled IoV networks. *J. Commun. Netw.* **25** (2), 197–207. https://doi.org/10.23919/JCN.2023.000004 (2023).
9. Waheed, A. et al. A comprehensive review of computing paradigms, enabling computation offloading and task execution in vehicular networks. *IEEE Access* **10**, 3580–3600. https://doi.org/10.1109/ACCESS.2021.3138219 (2022).
10. Rosmaninho, R., Raposo, D., Rito, P. & Sargento, S. Edge-cloud continuum orchestration of critical services: a smart-city approach. *IEEE Trans. Serv. Comput.* **18** (3), 1381–1396. https://doi.org/10.1109/TSC.2025.3568251 (May-June 2025).
11. Moghaddasi, K., Rajabi, S. & Gharehchopogh, F. S. Multi-objective secure task offloading strategy for blockchain-enabled IoV-MEC Systems: A Double Deep Q-Network Approach. *IEEE Access* **12**, 3437–3463. https://doi.org/10.1109/ACCESS.2023.3348513 (2024).
12. Khaled Hejja, S., Berri & Houda Labiod. Network slicing with load-balancing for task offloading in vehicular edge computing. *Elsevier* **34**, 1–16. https://doi.org/10.1016/j.vehcom.2021.100419 (2022).
13. Vemireddy, S. & Rout, R. R. Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing. *Comput. Netw.* **199**, 108463. https://doi.org/10.1016/j.comnet.2021.108463 (2021).
14. Phung, K.-H. et al. oneVFC—A Vehicular Fog Computation Platform for Artificial Intelligence in Internet of Vehicles. *IEEE Access* **9**, 117456–117470. https://doi.org/10.1109/access.2021.3106284 (2021).
15. Wu, C., Huang, Z. & Zou, Y. Delay Constrained Hybrid Task Offloading of Internet of Vehicle: A Deep Reinforcement Learning Method. *IEEE Access* **10**, 102778–102788. https://doi.org/10.1109/ACCESS.2022.3206359 (2022).
16. Jin, H., Gregory, M. A. & Li, S. A Review of Intelligent Computation Offloading in Multiaccess Edge Computing. *IEEE Access* **10**, 71481–71495. https://doi.org/10.1109/ACCESS.2022.3187701 (2022).
17. Chen, Z. et al. Mobility-Aware Seamless Service Migration and Resource Allocation in Multi-Edge IoV Systems. *IEEE Transactions on Mobile Computing* **24**(7), 6315–6332. https://doi.org/10.1109/TMC.2025.3540407 (2025).
18. Belal Ali, M. A., Gregory, S., Li & Omar Amjad Dib. Implementing zero trust security with dual fuzzy methodology for trust-aware authentication and task offloading in multi-access edge computing. *Elsevier* **241**, 1–12. https://doi.org/10.1016/j.comnet.2024.110197 (2024).
19. Asensio-Garriga, R. et al. Zsm-based e2e security slice management for ddos attack protection in mec-enabled v2x environments. *IEEE Open Journal of Vehicular Technology* **5**, 485–495. https://doi.org/10.1109/OJVT.2024.3375448 (2024).
20. Ye, W., Zheng, K., Wang, Y. & Tang, Y. Federated Double Deep Q-Learning-Based Computation Offloading in Mobility-Aware Vehicle Clusters. *IEEE Access* **11**, 114475–114488. https://doi.org/10.1109/ACCESS.2023.3324718 (2023).
21. Tran-Dang, H., Bhardwaj, S., Rahim, T., Musaddiq, A. & Kim, D.-S. Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues. *J. Commun. Netw.* **24**(1), 83–98. https://doi.org/10.23919/JCN.2021.000041 (2022).
22. Lin, B. et al. Computation offloading strategy based on deep reinforcement learning for connected and autonomous vehicle in vehicular edge computing. *Journal of Cloud Computing* https://doi.org/10.1186/s13677-021-00246-6 (2021).
23. Li, Z., Gong, J., Xiong, X. & Wang, D. Multi-slot Secure Offloading and Resource Management in VEC Networks: A Deep Reinforcement Learning-Based Method. *IEEE Access* **13**, 4533–4546. https://doi.org/10.1109/ACCESS.2024.3524636 (2025).
24. Rizwan, S., Husnain, G., Aadil, F., Ali, F. & Lim, S. Mobile Edge-Based Information-Centric Network for Emergency Messages Dissemination in Internet of Vehicles: A Deep Learning Approach. *IEEE Access* **11**, 62574–62590. https://doi.org/10.1109/ACCESS.2023.3288420 (2023).
25. Mustafa, A. S., Yussof, S. & Radzi, N. A. M. Multi-Objective Simulated Annealing for Efficient Task Allocation in UAV-Assisted Edge Computing for Smart City Traffic Management. *IEEE Access* **13**, 24251–24275. https://doi.org/10.1109/ACCESS.2025.3538676 (2025).
26. Mustafa, A. S., Yussof, S. & Radzi, N. A. M. CPFT-MOSA: A Comprehensive Parallel Fault-Tolerant Multi-Objective Simulated Annealing Framework for UAV-Assisted Edge Computing in Smart City Traffic Management. *IEEE Access* **13**, 66646–66673. https://doi.org/10.1109/ACCESS.2025.3558851 (2025).
27. Haq, S. A. U., Imran, M., Shah, N. & Muntean, G.-M. SDN-Based Edge Computing in Vehicular Communication Networks: A Survey of Existing Approaches. *IEEE Access* **13**, 74252–74287. https://doi.org/10.1109/ACCESS.2025.3561083 (2025).

28. Saad, M. M., Jamshed, M. A., Adedamola, A. I., Nauman, A. & Kim, D. Twin delayed DDPG (TD3)-based edge server selection for 5G-enabled industrial and C-ITS applications. *IEEE Open J. Commun. Soc.* **6**, 3332–3343. https://doi.org/10.1109/OJCOMS.2025.3545566 (2025).
29. Baktayan, A. A., Zahary, A. T. & Al-Baltah, I. A. A Systematic Mapping Study of UAV-Enabled Mobile Edge Computing for Task Offloading. *IEEE Access* **12**, 101936–101970. https://doi.org/10.1109/ACCESS.2024.3431922 (2024).
30. Cheng, Y. et al. Vehicular Fog Resource Allocation Approach for VANETs Based on Deep Adaptive Reinforcement Learning Combined With Heuristic Information. *IEEE Access* **12**, 139056–139075. https://doi.org/10.1109/ACCESS.2024.3455168 (2024).
31. Liu, G. et al. A collaborative computation and dependency-aware task offloading method for vehicular edge computing: A reinforcement le. *Springer* **11**(68), 1–15. https://doi.org/10.1186/s13677-022-00340-3 (2022).
32. Zhu, F. et al. Delay-Effective Task Offloading Technology in Internet of Vehicles: From the Perspective of the Vehicle Platooning. *IEEE Trans. Commun.* **73** (6), 3833–3848. https://doi.org/10.1109/TCOMM.2024.3493816 (2025).
33. Sinthia, A. K., Mahbub, N. I. & Huh, E.-N. Optimizing proactive content caching with mobility aware deep reinforcement & asynchronous federate learning in VEC. *ICT Express* **11** (2), 293–298. https://doi.org/10.1016/j.icte.2024.11.006 (2025).
34. Manzoor, S., Shakir, M. Z., Hasna, M. O. & Qaraqe, K. A. Mobility-aware federated learning-based proactive UAVs placement in emerging cellular networks. *IEEE Trans. Mach. Learn. Commun. Netw.* **2**, 1305–1318. https://doi.org/10.1109/TMLCN.2024.3439289 (2024).
35. Ballotta, L. et al. VREM-FL: Mobility-Aware Computation-Scheduling Co-Design for Vehicular Federated Learning. *IEEE Trans. Vehi. Technol.* **74** (2), 3311–3326. https://doi.org/10.1109/TVT.2024.3479780 (2025).
36. Alam, M., Matam, R. & Barbhuiya, F. A. OptFog: Optimized Mobility-Aware Task Offloading and Migration Model for Fog Networks, *2023 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Jaipur, India, 2023, pp. 539–544, https://doi.org/10.1109/ANTS59832.2023.10469215 (2023).
37. Ostrowski, K., Małecki, K., Dziurzański, P. & Singh, A. K. Mobility-aware multi-task migration and offloading scheme for Internet of Vehicles. *Elsevier* **219**, 1–24. https://doi.org/10.1016/j.jnca.2023.103724 (2023).
38. Liu, L. et al. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE* **24**(2), 2169–2182. https://doi.org/10.1109/TITS.2022.3142566 (2022).
39. Guan, S. & Boukerche, A. A Novel Mobility-aware Offloading Management Scheme in Sustainable Multi-access Edge Computing. *IEEE Trans. Sustainable Comput.* 1–1. https://doi.org/10.1109/tsusc.2021.3065310 (2021).
40. Parvini, M., Schulz, P. & Fettweis, G. Resource allocation in V2X networks: From classical optimization to machine learning-based solutions. *IEEE Open J. Commun. Soc.* **5**, 1958–1974. https://doi.org/10.1109/OJCOMS.2024.3380509 (2024).
41. Huang, Y. Multimedia tasks-oriented edge computing offloading scheme based on graph neural network in vehicular networks. *IEEE Access* **13**, 9780–9791. https://doi.org/10.1109/ACCESS.2025.3526627 (2025).
42. Wang, X., He, C., Jiang, W., Wang, W. & Liu, X. Generative AI based dependency-aware task offloading and resource allocation for UAV-assisted IoV. *IEEE* **6**, 3932–3949. https://doi.org/10.1109/OJCOMS.2025.3562720 (2025).
43. Talebkhah, M., Gordan, M., Sali, A., Khodamoradi, V. & Khodadadi, T. Task offloading for edge-IoV networks in the industry 4.0 era and beyond: A high-level view. *Elsevier* **54**, 1–40. https://doi.org/10.1016/j.jestch.2024.101699 (2024).
44. Ji, M. et al. Graph Neural Networks and Deep Reinforcement Learning-Based Resource Allocation for V2X Communications. *IEEE Int. Things J.* **12** (4), 3613–3628. https://doi.org/10.1109/JIOT.2024.3469547 (2025).
45. Annu & Rajalakshmi, P. Towards 6G V2X Sidelink: Survey of Resource Allocation—Mathematical Formulations, Challenges, and Proposed Solutions. *IEEE Open. J. Veh. Technol.* **5**, 344–383. https://doi.org/10.1109/OJVT.2024.3368240 (2024).
46. Liu, C. et al. Dependency-aware online task offloading based on deep reinforcement learning for IoV. *Springer* **13**(136), 1–17. https://doi.org/10.1186/s13677-024-00701-0 (2024).
47. Manzoor Ahmed, S. et al. Deep reinforcement learning approach for multi-hop task offloading in vehicular edge computing. *Elsevier* **59**, 1–11. https://doi.org/10.1016/j.jestch.2024.101854 (2024).
48. Song, X., Chen, Q., Wang, S. & Song, T. Cross-domain resources optimization for hybrid edge computing networks: Federated DRL approach. *Elsevier* 1–12. https://doi.org/10.1016/j.dcan.2024.03.006 (2024).
49. Shinde, S. S. & Tarchi, D. Multi-time-scale Markov decision process for joint service placement, network selection, and computation offloading in aerial IoV scenarios. *IEEE Trans. Netw. Sci. Eng.* **11**(6), 5364–5379. https://doi.org/10.1109/TNSE.2024.3445890 (2024).
50. Hao, H., Xu, C., Zhang, W., Yang, S. & Muntean, G.-M. Computing Offloading With Fairness Guarantee: A Deep Reinforcement Learning Method. *IEEE Trans. Circuits and Syst.Video Technol.* **33** (10), 6117–6130. https://doi.org/10.1109/TCSVT.2023.3255229 (2023).
51. Wu, Q. et al. Mobility-Aware Cooperative Caching in Vehicular Edge Computing Based on Asynchronous Federated and Deep Reinforcement Learning. *IEEE J. Selec. Topics Signal Process.* **17** (1), 66–81. https://doi.org/10.1109/JSTSP.2022.3221271 (2023).
52. Nguyen, L.-H., Nguyen, V.-L. & Kuo, J.-J. Efficient Reinforcement Learning-Based Transmission Control for Mitigating Channel Congestion in 5G V2X Sidelink. *IEEE Access* **10**, 62268–62281. https://doi.org/10.1109/ACCESS.2022.3182021 (2022).
53. Wu, J. et al. Resource allocation for delay-sensitive vehicle-to-multi-edges (V2Es) communications in vehicular networks: A multi-agent deep reinforcement learning approach. *IEEE Trans. Netw. Sci. Eng.* **8**(2), 1873–1886. https://doi.org/10.1109/tnse.2021.3075530 (2021).
54. Kuanishbay Sadatdiynov, L. et al. A review of optimization methods for computation offloading in edge computing networks. *Elsevier* **9** (2), 450–461. https://doi.org/10.1016/j.dcan.2022.03.003 (2024).
55. Aung, P. S., Nguyen, L. X., Tun, Y. K., Han, Z. & Hong, C. S. Deep reinforcement learning-based joint spectrum allocation and configuration design for STAR-RIS-assisted V2X communications. *IEEE Internet Things J.* **11**(7), 11298–11311. https://doi.org/10.1109/JIOT.2023.3329893 (2023).
56. Lee, I. & Kim, D. K. Decentralized multi-agent DQN-based resource allocation for heterogeneous traffic in V2X communications. *IEEE Access* **12**, 3070–3084. https://doi.org/10.1109/ACCESS.2023.3349350 (2024).
57. Noman, H. M. F. et al. Machine Learning Empowered Emerging Wireless Networks in 6G: Recent Advancements, Challenges and Future Trends. *IEEE Access* **11**, 83017–83051. https://doi.org/10.1109/ACCESS.2023.3302250 (2023).
58. Liu, J. et al. RL/DRL Meets Vehicular Task Offloading Using Edge and Vehicular Cloudlet: A Survey. *IEEE Internet of Things Journal* **9**(11), 8315–8338. https://doi.org/10.1109/JIOT.2022.3155667 (2022).
59. Chatterjee, T., Karmakar, R., Kaddoum, G., Chattopadhyay, S. & Chakraborty, S. A Survey of VANET/V2X Routing From the Perspective of Non-Learning- and Learning-Based Approaches. *IEEE Access* **10**, 23022–23050. https://doi.org/10.1109/ACCESS.2022.3152767 (2022).
60. Wang, J. & Wang, L. Mobile edge computing task distribution and offloading algorithm based on deep reinforcement learning in internet of vehicles. *J. Ambient Intell. Humaniz. Comput.* https://doi.org/10.1007/s12652-021-03458-5 (2021).
61. Zhang, Z. et al. DRL-Based Optimization for AoI and Energy Consumption in C-V2X Enabled IoV. *IEEE.*, pp.1–16. https://doi.org/10.1109/TGCN.2025.3531902 (2025).
62. Elgendy, I. A. et al. Optimizing Energy Efficiency in Vehicular Edge-Cloud Networks Through Deep Reinforcement Learning-Based Computation Offloading. *IEEE* **12**, 191537–191550. https://doi.org/10.1109/ACCESS.2024.3514881 (2024).
63. Li, X. A Computing Offloading Resource Allocation Scheme Using Deep Reinforcement Learning in Mobile Edge Computing Systems. *Journal of Grid Computing* https://doi.org/10.1007/s10723-021-09568-w (2021).
64. Luo, Q., Li, C., Luan, T. H., Shi, W. & Wu, W. Self-learning based computation offloading for internet of vehicles: Model and algorithm. *IEEE Trans. Wireless Commun.* **20**(9), 5913–5925. https://doi.org/10.1109/twc.2021.3071248 (2021).

65. Hsu, C. H., Chiang, Y., Zhang, Y. & Wei, H. Y. Mobility-Aware QoS Promotion and Load Balancing in MEC-Based Vehicular Networks: A Deep Learning Approach. *2021 IEEE 93rd Veh. Technol. Conf. (VTC2021-Spring)*. https://doi.org/10.1109/vtc2021-spring51267.2021.9448705 (2021).
66. Labriji, I. et al. Mobility aware and dynamic migration of MEC services for the Internet of Vehicles. *IEEE Trans. Netw. Serv. Manag.* **18**(1), 570–584. https://doi.org/10.1109/tnsm.2021.3052808 (2021).
67. Liao, Z., Ma, Y., Huang, J., Wang, J. & Wang, J. HOTSPOT: A UAV-assisted dynamic mobility-aware offloading for mobile-edge computing in 3-D space. *IEEE Internet Things J.* **8**(13), 10940–10952. https://doi.org/10.1109/jiot.2021.3051214 (2021).
68. Zhang, K., Cao, J., Maharjan, S. & Zhang, Y. Digital Twin Empowered Content Caching in Social-Aware Vehicular Edge Networks. *IEEE Trans. Comput. Social Syst.* 1–13. https://doi.org/10.1109/tcss.2021.3068369 (2021).
69. Farhangi Maleki, E., Mashayekhy, L. & Nabavinejad, S. M. Mobility-Aware Computation Offloading in Edge Computing using Machine Learning. *IEEE Trans. Mob. Comput.* 1–1. https://doi.org/10.1109/tmc.2021.3085527 (2021).
70. Li, Z., Kong, Y., Wang, C. & Jiang, C. DDoS Mitigation Based on Space-Time Flow Regularities in IoV: A Feature Adaption Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* 1–17. https://doi.org/10.1109/tits.2021.3066404 (2021).
71. Lin, P., Song, Q., Yu, F. R., Wang, D. & Guo, L. Task Offloading for Wireless VR-Enabled Medical Treatment with Blockchain Security Using Collective Reinforcement Learning. *IEEE Internet Things J.* https://doi.org/10.1109/jiot.2021.3051419 (2021).
72. Andr´ e-Luc Beylot & Kacimi, R. Mobile edge computing for V2X architectures and applications: A survey. *Elsevier* **206**, 1–24. https://doi.org/10.1016/j.comnet.2022.108797 (2022).
73. Anokye, S., Ayepah-Mensah, D., Seid, A. M., Boateng, G. O. & Sun, G. Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks. *IEEE Syst. J.* 1–12. https://doi.org/10.1109/jsyst.2021.3082837 (2021).
74. Dziyauddin, R. A. et al. Computation offloading and content caching and delivery in vehicular edge network: A survey. *Comput. Netw.* **197**, 108228. https://doi.org/10.1016/j.comnet.2021.108228 (2021).
75. Xie, B. et al. MOB-FL: Mobility-Aware Federated Learning for Intelligent Connected Vehicles, *ICC 2023 - IEEE International Conference on Communications*, Rome, Italy, pp. 3951–3957, https://doi.org/10.1109/ICC45041.2023.10279773 (2023).
76. Chaowei, W. et al. Collaborative Caching in Vehicular Edge Network Assisted by Cell-Free Massive MIMO. *Chin. J. Electron.* **32**(6), 1218–1229. https://doi.org/10.23919/cje.2022.00.294 (2023).
77. Safavifar, Z., Mechalikh, C., Xie, J. & Golpayegani, F. Enhancing VRUs Safety Through Mobility-Aware Workload Orchestration with Trajectory Prediction using Reinforcement Learning, *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, Bilbao, Spain, 2023, pp. 6132–6137, https://doi.org/10.1109/ITSC57777.2023.10421846 (2023).
78. Han, Y., Li, X. & Zhou, Z. Dynamic Task Offloading and Service Migration Optimization in Edge Networks. *Int. J. Crowd Sci.* **7**(1), 16–23. https://doi.org/10.26599/IJCS.2022.9100031 (2023).
79. Jeon, Y., Baek, H. & Pack, S. Mobility-aware optimal task offloading in distributed edge computing. *2021 Int. Conf. Inform. Netw. (ICOIN)* https://doi.org/10.1109/icoin50884.2021.9334008 (2021).
80. Zhang, Y., Zhang, M., Fan, C., Li, F. & Li, B. Computing resource allocation scheme of IOV using deep reinforcement learning in edge computing environment. *EURASIP J. Adv. Signal Process.* https://doi.org/10.1186/s13634-021-00750-6 (2021).
81. Du, J., Zhao, L., Sun, Y. & Wang, J. Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization. *IEEE Trans. Veh. Technol.* **67**(11), 11086–11097. https://doi.org/10.1109/TVT.2018.2881191 (2018).
82. Riley, G. F. & Henderson, T. R. *The ns-3 Network Simulator* In (eds Wehrle, K. et al.) (2010).
83. García, F., Camachuelo, R. & Fernández, E. *ns3-gym: Extending OpenAI Gym for Networking Research*. In: *Proceedings of the 10th International Conference on Network and Service Management (CNSM)*. IEEE, pp.1–5. https://doi.org/10.1109/CNSM.2018.8585029 (2018).

Author contributions

All authors contributed to the study's conception and design. Material preparation, data collection, and analysis were performed by **Sadineni Neelima, S. Rama Sree, N. Ramakrishnaiah**.**^{}**The first draft of the manuscript was written by **Sadineni Neelima** all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding

No financial support was received by the authors in this research.

Declarations

Competing interests

The authors declare no competing interests.

Consent for publication

The authors give consent for their publication.

Ethical approval

This research does not involve humans or animals, so no ethical approval is required.

Additional information

Correspondence and requests for materials should be addressed to S.N.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.