Towards Generalizable RL-Based Task Offloading for Vehicular Edge Computing

Agon Memedi*, Negin Masoudifar*, Torsten Braun†, and Falko Dressler*
*School of Electrical Engineering and Computer Science, TU Berlin, Germany
†Institute of Computer Science, University of Bern, Bern, Switzerland
{memedi,negin.masoudifar,dressler}@ccs-labs.org, torsten.braun@unibe.ch

***Abstract*—Deep reinforcement learning (DRL) has been used very successfully for a variety of applications in the field of vehicular networking. We investigate the generalizability of a DRL-based scheduling approach for task offloading in vehicular edge computing (VEC). We propose a hybrid scheduling policy that combines heuristic task prioritization with a learned DQN-based resource selection. Two deep Q-learning (DQN) architectures are compared: a conventional multilayer perceptron (MLP) with fixed-size inputs; and a permutation-invariant architecture that uses a lightweight transformer encoder. Trained exclusively on a simple straight-road scenario, the permutation-invariant DQN achieves perfect selection of the fastest processing vehicle when tested in an unseen realistic scenario. In contrast, the MLP-based DQN suffers severe degradation due to positional bias and padding artifacts. Results demonstrate that set-based, permutation-invariant representations are essential for sample efficiency and generalization in VEC environments with high variability.**

I. Introduction

The recent surge of computation-intensive applications such as autonomous driving, generative AI, and extended reality (XR) services has significantly increased the demand for low-latency and high-throughput computing and networking resources [1]. Initially, traditional distributed computing architectures like cloud and edge computing have been instrumental in addressing this demand [2]. Nevertheless, their scalability is often constrained by inherent limitations, including high latency in cloud environments and deployment and maintenance costs in edge infrastructures [3]. As a result, research has shifted toward novel, more dynamic, distributed computing paradigms that can leverage available computing resources more opportunistically.

In this context, vehicular edge computing (VEC) has emerged as a promising solution that aims to utilize on-board vehicle resources and nearby edge infrastructure to perform distributed processing of computational tasks [4]. By grouping vehicles into clusters known as vehicular micro cloud (VMC) [5], vehicles can collaboratively share, offload, and process computational workloads. Therefore, vehicles do not only consume services but also act as mobile compute nodes capable of providing services. This paradigm enables scalable, on-demand computing that extends beyond static cloud infrastructure, effectively transforming vehicles into mobile edge nodes [6].

However, the inherent mobility and spatiotemporal dynamics of the road traffic lead to highly variable resource availability and communication conditions. This variability complicates task scheduling and resource management, as it requires systems to adapt continuously to changing network topologies, link qualities, and processing capacities [6]. This shift in paradigm, where demand for resources and their availability are both dynamic and unpredictable, requires the development of advanced resource management strategies that can operate effectively in such environments. Traditional optimization-based methods often fail to meet these requirements, as they rely on static approaches that do not hold in realistic vehicular environments [7].

Machine learning (ML), and particularly deep reinforcement learning (DRL), has been widely adopted as a means to address these challenges [8], [9], [10], [11], [12]. By learning policies from historical and real-time data, DRL-based agents can adapt to complex and non-stationary vehicular environments, providing flexible and efficient decision-making for task offloading and scheduling. Nevertheless, many of the works in this domain typically utilize fully connected (dense) neural architectures, such as multilayer perceptron (MLP), as function approximators for DRL algorithms. While effective in small-scale or stationary settings, such architectures struggle to generalize across diverse vehicular contexts, where system dynamics vary continuously.

To address this gap, this paper investigates the generalization capabilities of RL-based task scheduling in VEC. Specifically, we build upon our previous work based on deep Q-learning (DQN) [13] with dense neural architecture, and introduce a transformer-enhanced model that integrates self-attention mechanisms. Our objective is to enhance policy transferability and robustness, enabling RL agents trained under specific scenarios to generalize effectively to unseen configurations involving different vehicular densities, mobility patterns, and communication conditions.

Our contributions can be summarized as follows:

- We propose a transformer-enhanced reinforcement learning (RL) framework that leverages self-attention mechanisms, enabling agents trained in simplified environments to generalize effectively when deployed in more complex scenarios.
- We conduct, to the best of our knowledge, the first evaluation of RL generalization in realistic VEC scenarios.
- Our experiments provide first insights into the generalization behavior of attention-based RL architectures under different road traffic and computational load dynamics.

II. Related Work

ML-based approaches, particularly DRL, have been extensively explored for resource management in VEC, including computation offloading [7], [8], [11], [12], communication resource allocation [9], and content caching [10].

Liu et al. [8] proposed a cooperative task offloading framework to address resource utilization, latency and energy consumption in mobile edge computing (MEC). Their method combines multi-agent asynchronous DRL (i.e., MADQN and MADDPG) for discrete server selection and continuous resource allocation, respectively. Additionally, an offline-trained transformer model is used to predict future task arrival times and computational demands.

Similarly, Han et al. [7] propose the Distributed Transformer-based Actor-Critic (DTAC) algorithm, that integrates the transformer architecture into an actor-critic framework to manage a hybrid high-dimensional action space consisting of discrete offloading decisions and continuous resource allocation. The paper introduces a decentralized transfer learning (TL) method to mitigate the signaling overhead inherent in centralized approaches. This approach adapts a converged centralized model into a distributed framework where individual user equipments (UEs) can independently manage resources based solely on local information.

Geng et al. [11] propose a multi-agent DRL framework for VEC, where idle vehicular resources act as temporary edge nodes. Their approach incorporates convolutional and recurrent layers to process spatio-temporal features, while tasks are modeled as a directed acyclic graph (DAG) to capture complex subtask dependencies. Li et al. [12] focus on MEC systems where large-scale information exchange incurs significant overhead. To address this, the authors propose a distributed cooperative RL algorithm that integrates an attention mechanism to identify active users and allocate resources accordingly, along with a communication module that shares action intentions among dynamically formed user groups to reduce long-term task delay.

Ju et al. [9] propose a computation offloading scheme for VEC environments that jointly considers processing delay and communication security under spectrum sharing constraints. In their system, vehicles offload computation tasks to a roadside mobile server while reusing spectrum allocated to vehicle-to-vehicle (V2V) communication for vehicle-to-infrastructure (V2I) transmission. Security is ensured through physical layer security mechanisms. The joint optimization of spectrum access selection, transmit power control, and computation resource allocation is formulated as a multi-agent collaborative decision process and solved using a multi-agent DRL approach based on the double DQN (DDQN) algorithm. Each vehicle acts as an autonomous agent that locally observes channel conditions, interference levels, computation resource availability, and task characteristics, and selects a joint action accordingly.

In [10], the authors propose a content caching–assisted VEC framework aimed at mitigating repeated task offloading and enabling the reuse of popular computation results. To address challenges arising from irregular vehicular network topologies and uncertain environmental dynamics, they introduce a multi-agent RL scheme based on graph attention mechanisms. By integrating features from neighboring nodes, the proposed approach captures spatial relationships among vehicles and enhances cooperative caching decisions in dynamic VEC environments.

Overall, while these studies demonstrate the effectiveness of DRL for adaptive resource management in VEC, they largely overlook the complexity of training and deploying learning-based solutions in highly dynamic settings. Most evaluations are conducted in simplified or homogeneous scenarios, with limited discussion of scalability, convergence behavior, or policy generalization across heterogeneous traffic conditions and network topologies. Given the inherent mobility and volatility of vehicular edge environments, efficient training and robust generalization of RL-based policies remain critical and underexplored challenges.

III. System Model

Conceptually, we follow the system model for task offloading described in [14]. In our setup, we consider a VEC environment composed of a dynamic set of vehicles denoted by $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$, where each vehicle $v_i \in \mathcal{V}$ is equipped with an on-board computational unit, such as a CPU or GPU, which we characterize by its processing capacity rather than its specific architecture. Vehicles follow realistic mobility patterns and may dynamically enter or leave the scenario as traffic conditions evolve. To enable localized computation and task offloading, vehicles are organized into VMCs, which act as distributed edge computing infrastructures close to where data and computation requests are generated.

A. Vehicular Microcloud (VMC)

A VMC is defined as a logical cluster of nearby vehicles that collaboratively provide computation, storage [15] and communication resources within a localized area (see Figure 1). We assume that VMCs exist readily or form spontaneously in areas with high vehicle density, such as intersections, parking lots, or charging stations, where vehicles naturally accumulate or remain for extended periods. Although individual vehicles

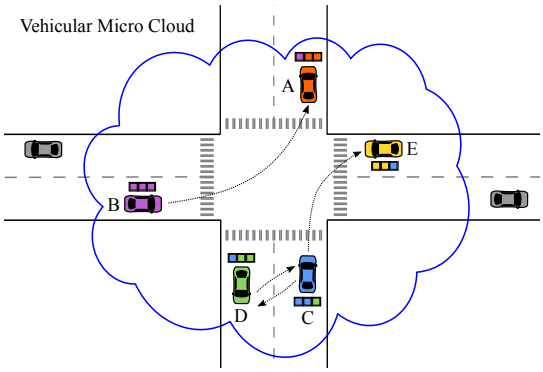


Figure 1. Task offloading in a VMC. Vehicles can generate, exchange and process tasks.

are mobile, the VMC itself is spatially static and anchored to its corresponding region of interest; however, it evolves dynamically as members join or leave.

To facilitate collaboration within a VMC m, the set of vehicles $\mathcal{V}_m \subseteq \mathcal{V}$ need to communicate with each other (via V2X protocols such as IEEE 802.11bd [16] and C-V2X [17]) to enable task offloading and cooperative execution. For the purpose of this study, however, we abstract away the underlying channel dynamics and assume seamless one-hop communication among the vehicles within a VMC.

B. Computation Model

1) Vehicles: Each *vehicle* functions as both a *task producer* and a *task consumer*. Specifically, a vehicle can generate computational tasks that must be processed within a deadline, while also serving as a computational node capable of executing tasks locally or handling tasks offloaded by neighboring vehicles within the same VMC. The computation power of a vehicle is characterized by its processing capacity P_i. At any given time, a vehicle is either *idle* (i.e., available to execute a new task) or *busy* (i.e., currently processing a task). Each vehicle has a processing queue with a capacity of one, allowing it to execute only a single task at a time. The set of idle vehicles at time t is denoted by

$$\mathcal{V}_{\text{idle}}(t) = \{v_i \in \mathcal{V} \mid v_i \text{ is not processing a task at time } t\}. \tag{1}$$

Vehicles generate tasks periodically but with heterogeneous rates. Specifically, for each vehicle, the inter-arrival time between two consecutive tasks is fixed and equal to T_{gap}, where T_{gap} is sampled once at initialization from a distribution defined in the corresponding workload (cf. Section V-B).

2) Tasks: A *task* τ_j is represented by the tuple:

$$\tau_j = \{A_j, D_j, \kappa_j, s_j\}, \tag{2}$$

where A_j is the arrival time, D_j is the deadline, κ_j denotes the computational complexity (in required CPU cycles), and s_j identifies the source vehicle. Tasks can either be executed locally by their originating vehicle or offloaded to idle vehicles within the same VMC: $v_i \in \mathcal{V}_m$. Tasks are non-preemptible once execution begins, and since vehicles have a processing queue of capacity one, new tasks must wait until the vehicle becomes idle. The execution time required by vehicle v_i to process task τ_j is given by:

$$T_{ij}^{\text{proc}} = \frac{\kappa_j}{P_i}. \tag{3}$$

Tasks that exceed their deadlines ($t > A_j + D_j$) are discarded.

C. Scheduler and Policies

A centralized *scheduler* operates within the VMC, assuming a logical central entity (e.g., a lead vehicle or roadside unit) with global visibility over all idle vehicles $\mathcal{V}_{\text{idle}}$ and pending tasks $\mathcal{T}_{\text{active}}$. This entity continuously monitors the system state at discrete simulation steps and is responsible for matching computational tasks to idle vehicles to optimize performance objectives, such as maximizing the number of tasks completed before their deadlines or minimizing average task latency. The short-term stability of communication links and vehicle presence enables the scheduler to assume quasi-stationary network conditions when optimizing task-vehicle assignments. At each decision step t, the scheduler observes the current system state s_t, which includes the set of idle vehicles $\mathcal{V}_{\text{idle}} \subseteq \mathcal{V}_t$ and the set of active (unassigned) tasks $\mathcal{T}_{\text{active}} \subseteq \mathcal{T}_t$, where $\mathcal{V}_t$ and $\mathcal{T}_t$ denote the vehicles and tasks present in the VMC at time t, respectively.

The scheduler implements a policy $\pi : \mathcal{S} \to \mathcal{A}$ that selects an action a_t based on the current system state s_t. Thus, the scheduling decision at time t is given by

$$a_t = \pi(s_t). \tag{4}$$

An action $a_t \in \mathcal{A}$ is defined as a tuple (v_i^*, τ_j^*), which corresponds to assigning an active task $\tau_j^* \in \mathcal{T}_{\text{active}}$ to an idle vehicle $v_i^* \in \mathcal{V}_{\text{idle}}$:

$$a_t = (v_i^*, \tau_j^*). \tag{5}$$

If no feasible assignment exists (e.g., no idle vehicles are available or all tasks have expired deadlines), the scheduler outputs $a_t = \emptyset$. We do not explicitly account for assignment costs (e.g., communication or energy overhead); scheduling decisions are evaluated solely through the defined reward function (cf. Section IV-3).

The scheduler implements two main classes of policies (cf. Section IV-A): (i) heuristic-based policies that rely on deterministic rules to prioritize assignments (e.g., earliest deadline first, lowest complexity, random task selection); and (ii) RL-based policies (e.g., DQN) that learn assignment of tasks and vehicles via repeated interaction with the environment, aiming to maximize long-term cumulative rewards. A hybrid variant, combining heuristic and RL policies is also possible.

IV. Reinforcement Learning Setup

To solve the dynamic task-vehicle assignment in a VMC, we model the problem as a Markov decision process (MDP) and and solve it using a DRL approach based on deep Q-learning. We implement a hybrid scheduling policy that decouples task prioritization from resource assignment. In the first stage, a heuristic (e.g., Earliest Deadline First) pre-selects a task from the task queue; then, the DQN agent performs the actual resource assignment.

1) State Space: At each decision step t, the scheduler observes the current system state s_t, which comprises the features of the selected task and the set of available (idle) vehicles. Formally, the state is represented as:

$$s_t = \left\{\mathbf{f}_\tau, \mathbf{F}_{\mathcal{V}_{\text{idle}}(t)}\right\}, \tag{6}$$

where:

- $\mathbf{f}_\tau \in \mathbb{R}^2$ is the feature vector of the current task $\tau_j = \{A_j, D_j, \kappa_j, s_j\}$:

$$\mathbf{f}_\tau = [\kappa_j, D_j]^\top.$$

- $\mathbf{F}_{\mathcal{V}_{\text{idle}}(t)} \in \mathbb{R}^{N_{\text{idle}} \times 1}$ is a matrix stacking the processing capacities of all idle vehicles:

$$\mathbf{F}_{\mathcal{V}_{\text{idle}}(t)} = \left[P_{i_1}, P_{i_2}, \ldots, P_{i_{N_{\text{idle}}}}\right]^\top,$$

where $N_{\text{idle}} = |\mathcal{V}_{\text{idle}}(t)|$.

The state space is heterogeneous and variable-sized due to the dynamic number of idle vehicles.

2) Action Space: The action space is discrete and time-varying. At each decision step t, after a task has been pre-selected, the agent must decide which idle vehicle (if any) should execute it.

Let $\mathcal{V}_{\text{idle}}(t) = \{v_1, v_2, \ldots, v_{N_{\text{idle}}(t)}\}$ be the set of currently idle vehicles with cardinality $N_{\text{idle}}(t)$. The agent can either assign the task to one of the idle vehicles, or output the null action (no assignment) when assignment is not beneficial.

Formally, the action space is defined as:

$$\mathcal{A}(t) = \{0, 1, \ldots, N_{\text{idle}}(t) - 1\} \cup \{\emptyset\}, \tag{7}$$

where action $a_t = k$ ($k = 0, \ldots, N_{\text{idle}}(t) - 1$) means assigning the current task to the k-th vehicle in the current ordering of $\mathcal{V}_{\text{idle}}(t)$, and $a_t = \emptyset$ means that the task is not assigned to any vehicle at this step.

3) Reward Function: The reward signal r_t is designed to promote timely and efficient task completion while penalizing suboptimal (or invalid) assignments. Let $v^* = v_{i_k}$ be the vehicle selected via action a_t, and let $T^{\text{comp}} = t + \frac{\kappa_j}{P_{i_k}}$ be the estimated completion time. The reward is defined as:

$$r_t = \begin{cases} 2.0, & \text{if } a_t = a^* \text{ and } T^{\text{comp}} \leq A_j + D_j, \\ 0.0, & \text{if } a_t = a^* \text{ but } T^{\text{comp}} > A_j + D_j, \\ -10.0, & \text{otherwise}, \end{cases} \tag{8}$$

where $a^* = \arg\max_{i \in \mathcal{V}_{\text{idle}}(t)} P_i$ is the oracle action selecting the vehicle with maximum processing power. The agent is penalized for selecting any vehicle that is not the fastest available, even if that vehicle would still complete the task before the deadline. The motivation behind this design is to guide the agent toward selecting the best available vehicle, as such resources may be limited. In vehicular microclouds, high-capacity vehicles are scarce and valuable shared assets; prioritizing their selection enables more efficient overall system utilization.

4) Transition Dynamics: Upon executing action a_t, the environment transitions as follows:

- The assigned task τ_j is removed from the pending set $\mathcal{T}_{\text{active}}$.
- The selected vehicle becomes busy until completion (queue capacity = 1).
- Time advances to the next decision step (depending on task arrival or vehicle availability).
- The episode terminates when $t = T_{\text{duration}}$, the simulation horizon.

The next state s_{t+1} reflects updated idle vehicle set and the next heuristically pre-selected task.

Table I
COMPARISON OF THE TWO DQN IMPLEMENTATIONS

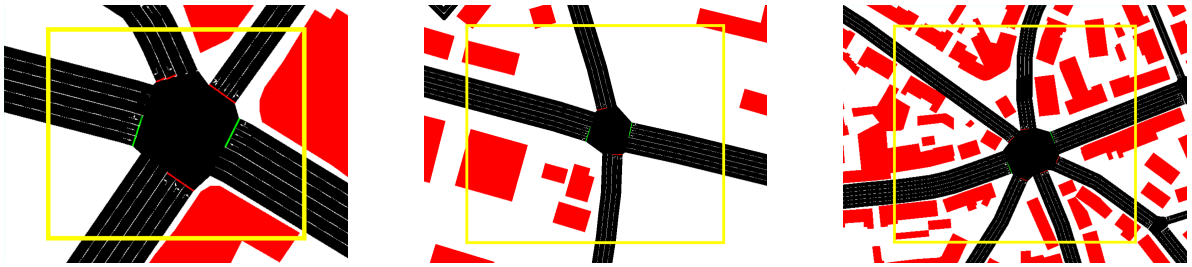
Feature	MLP-DQN	PI-DQN
Observation space	Fixed-length vector	Dictionary + variable-length sequence
State encoding	$[n_{\text{idle}}, \hat{\kappa}_j, \hat{D}_j, \mathbf{P}]$	$\mathbf{f}_\tau \in \mathbb{R}^2$ and $\{P_i \mid v_i \in \mathcal{V}_{\text{idle}}\}$
Padding / masking	Zero-padding to $N_{\max}$	No padding (native sequences)
Network input size	Constant	Variable (N_{idle})
Architecture	Fully connected MLP with 2 hidden layers	Single-layer Transformer encoder + shared MLP
Action masking	Explicit masking ($Q = -\infty$)	Implicit (argmax over valid vehicles)

A. MLP-based DQN vs. Permutation-Invariant DQN

In the remainder of this work, we focus on the RL-based policies. Specifically, to study the generalization capability of the RL agent, we implement two DQN variants that share the same objective but differ in state representation and neural architecture.

1) MLP-DQN: For multilayer perceptron (MLP)-based DQN (MLP-DQN), we adopt our previous implementation [13]. The Q-network is implemented as a fully connected MLP with two hidden layers, and outputs Q-values for a fixed action space of size $N_{\max}$ (corresponding to an upper bound on the number of idle vehicles). Since fully connected MLP architectures require fixed-size input and output spaces, we employ a *padded state representation* and *action masking* to accommodate the variable number of available vehicles. Specifically, the state s is defined as a fixed-length vector: $s = [n_{idle}, \hat{\kappa}_j, \hat{D}_j, \mathbf{P}]$, where n_{idle} is the number of idle vehicles, $\hat{\kappa}_j$ and $\hat{D}_j$ are the normalized task features, and $\mathbf{P}$ is a vector of processing capacities P_i of idle vehicles, zero-padded up to $N_{\max}$. During action selection, invalid actions corresponding to padded or unavailable vehicle entries are masked by assigning their Q-values to $-\infty$, ensuring they are excluded from the maximization operation.

2) PI-DQN: For permutation-invariant DQN (PI-DQN) we adopt a set-based state representation that naturally accommodates a variable number of idle vehicles. Task features are first embedded into a latent space and replicated across all vehicle entries, then concatenated with each vehicle's processing capacity to form a set of task-vehicle tokens. These tokens are processed by a single-layer Transformer encoder with one attention head and model dimension $d_{\text{model}} = 256$, enabling context-aware interactions among vehicles while preserving permutation invariance. The resulting representations are passed through a shared MLP to produce one Q-value per idle vehicle, corresponding directly to the available actions. Since Q-values are computed only for existing vehicles, neither state padding nor explicit action masking is required. This architecture follows the *deep sets* principle, ensuring that the learned policy is invariant to permutations of vehicle ordering and it uses self-attention to model interactions among vehicles.

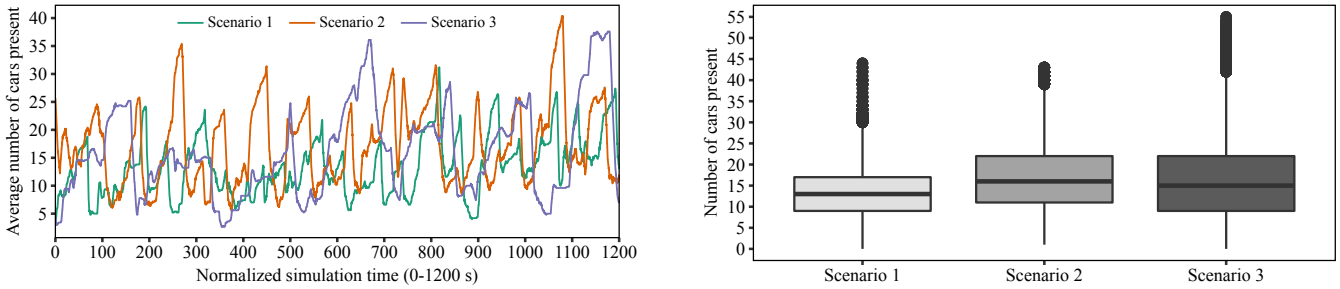


(a) Scenario 1: Five-way intersection; Approx. center: 49.6133°N, 6.1199°E); Bounding box: (6460,7190)–(6540,7255).

(b) Scenario 2: Complex four-leg intersection; Approx. center: 49.6164°N, 6.0984°E; Bounding box: (4865,7560)–(5040,7707).

(c) Scenario 3: Medium-sized roundabout; Approx. center 49.6064°N, 6.1063°E); Bounding box: (5390,6420)–(5580,6590).

Figure 2. Definition of the ROIs used as test scenarios, showing real-world coordinates (WGS84 standard) and bounding box definitions according to SUMO coordinates: $(min_x, min_y) - (max_x, max_y)$.



(a) Time series of cars in the scenarios.

(b) Car count distribution.

Figure 3. Learning performance of the RL agent with different DQN variants.

V. Evaluation

To evaluate the generalization capability of the proposed RL-based scheduler, we train both DQN variants (MLP-DQN and PI-DQN) on a simple synthetic scenario and test them *zero-shot* on three more complex, previously unseen real-world traffic scenarios with three distinct workload configurations. Our evaluation focuses on robustness to (i) changes in traffic topology and vehicle density and (ii) shifts in workload characteristics.

A. Training Scenario and Setup

The training scenario consists of a simple two-lane bidirectional straight road where vehicles travel at constant speed. At most 16 vehicles are simultaneously present, remaining below the capacity limit of $N_{\max} = 20$ used during training. This limit is selected as a conservative upper bound on the average number of idle vehicles observed across training and evaluation scenarios. It ensures a fixed action dimensionality for the MLP-DQN while safely exceeding the maximum vehicle count encountered during training. This setting induces a small and relatively stable VMC, allowing agents to learn an effective resource selection policy (according to the reward function, cf. Section IV-3) without exposure to large-scale variability. Computational tasks for the training scenario are generated with the distributions defined for the *training workload* in Table II. Both the MLP-DQN and PI-DQN agents are trained for 2000 episodes using an identical set of hyperparameters, summarized in Table III.

Table II
Parameter distributions for different workloads

Parameter	Training WL	Test WL
Task complexity (κ_j)	$\mathcal{U}(1,7)$	$\mathcal{U}(12,15)$
Task deadline (D_j)	$\mathcal{U}(1,6)$	$\mathcal{U}(2,6)$
Vehicle proc. power (P_i)	$\mathcal{U}(1,4)$	$\mathcal{U}(7,10)$
Task inter-arrival (T_{gap})	$\mathcal{U}(2,4)$	$\mathcal{U}(2,4)$ / $\mathcal{U}(6,8)^*$

Note: We also deploy Test WL (same as WL, but $T_{gap} \sim \mathcal{U}(6,8)$).

Table III
Training hyperparameters for the DRL agent

Hyperparameter	Value
Number of training episodes	2000
Replay buffer capacity	100,000
Batch size	32
Discount factor (γ)	0.99
Optimizer	Adam
Learning rate	10^{-3}
ϵ-greedy exploration	Exp. decay (1.0 to 0.1)
ϵ-decay rate	0.008
Target network update frequency	10 steps

B. Test Scenarios and Workload Configurations

We evaluate the agents' generalization ability across three different region of interests (ROIs) from the LuST scenario [18]. As shown in Figure 2, each test scenario features a distinct road topology and induces different traffic dynamics:

- **Scenario 1**: Five-way intersection simulated during morning rush hour (06:10). The region exhibits high vehicle

density and complex merging patterns (Figure 2a).

- **Scenario 2**: Complex four-leg intersection simulated during afternoon peak traffic (17:00), with significant congestion and turning movements (Figure 2b).
- **Scenario 3**: Medium-sized roundabout and adjacent roads, simulated under midday traffic conditions (12:00), featuring moderate flow and frequent lane changes (Figure 2c).

Figure 3 illustrates the traffic demand across scenarios; The number of vehicles per scenario is shown as a time series (Figure 3a), and as the aggregate distribution across multiple simulation runs for each scenario (Figure 3b).

For each scenario, we test the MLP-DQN and PI-DQN in different workload regimes (summarized in Table II):

- **Training WL (Workload 1):** Task complexity, deadlines, vehicle processing power, and task arrival rates follow the same distributions as used during training. This workload serves as a baseline and isolates the effect of increased traffic complexity and larger VMC sizes in unseen scenarios.
- **Test WL (Workload 2):** Tasks have higher computational complexity, with $\kappa_j \sim \mathcal{U}(12, 15)$, while vehicles have higher processing capabilities, with $P_i \sim \mathcal{U}(7, 10)$. Task arrival rates are identical to those used during training. This workload evaluates robustness to simultaneous shifts in task difficulty and resource capacity, representing a more demanding operating regime. By keeping the total number of tasks comparable to the training setup, this workload decouples workload intensity from task frequency, enabling a controlled analysis of how changes in task and vehicle characteristics alone affect the agents' decision-making behavior.
- **Test WL* (Workload 3):** Task complexity and vehicle processing power follow the high-demand distributions of Test WL, while tasks arrive less frequently due to longer inter-arrival times. Larger values of T_{gap} correspond to lower task generation rates and, thus, lower system load.

Note: the workload parameters are expressed as unit-less scalars to capture relative changes in task complexity (κ_j), task deadline (D_j), vehicle processing power (P_i), and task inter-arrival time (T_{gap}). This abstraction enables a clean evaluation of policy robustness under controlled distributional shifts, so the results are not influenced by details of particular hardware or exact timing.

Each configuration is evaluated over five independent simulation runs of 1200 s. To ensure statistical validity, vehicle mobility traces and workload parameters are randomized across different runs. All experiments are conducted with VE-SIM,[1] our open-source framework that integrates SimPy (discrete-event simulation), SUMO (microscopic road traffic simulation), and PyTorch (ML framework).

C. Results and Analysis

As shown in Figure 4, PI-DQN exhibits a better learning performance: it converges faster and achieves higher episodic reward (∼450) compared to MLP-DQN (∼100). The performance difference, even in the simple training environment, shows the limitation of (padded and) fixed-size representations: a significant fraction of the MLP network's capacity is spent on learning to ignore irrelevant entries, hindering sample efficiency and final policy quality.

[1]https://github.com/agnmmd/ve-sim

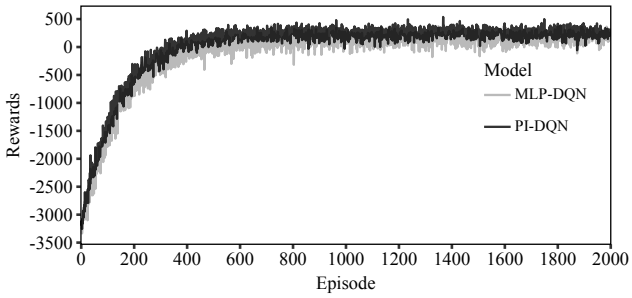
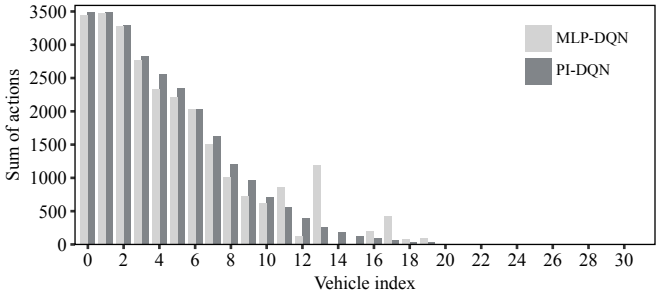
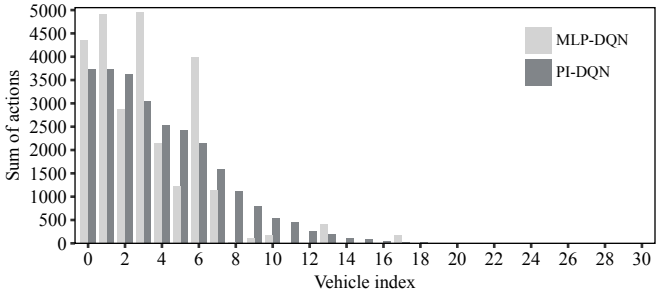


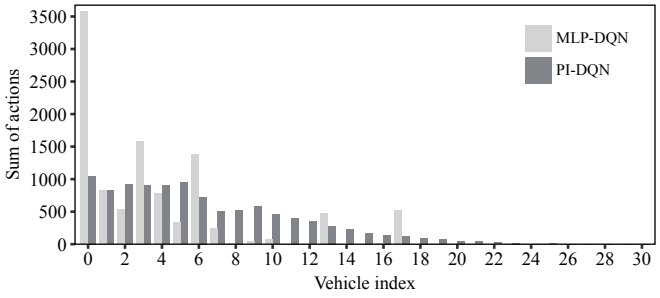
Figure 4. Learning performance of the RL agent with different DQN variants.



(a) Scenario 1, Training WL.



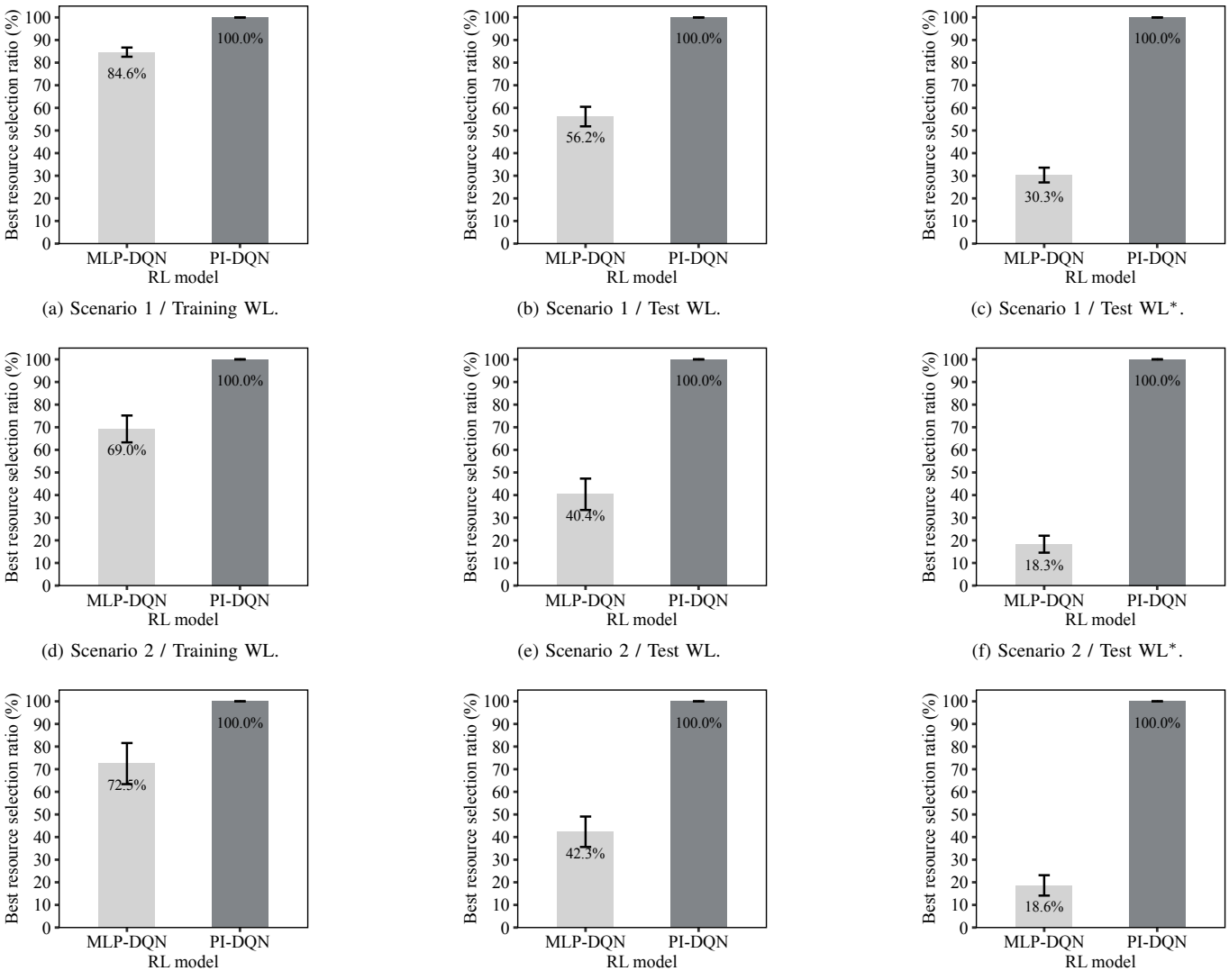
(b) Scenario 1, Test WL.



(c) Scenario 1, Test WL*.

Figure 5. Aggregate of agent's action selections over multiple simulation runs in Scenario 1 under training and test workloads.

Figure 5 illustrates the action selection patterns of both approaches across training and test workloads in Scenario 1.



(a) Scenario 1 / Training WL. (b) Scenario 1 / Test WL. (c) Scenario 1 / Test WL*.

(d) Scenario 2 / Training WL. (e) Scenario 2 / Test WL. (f) Scenario 2 / Test WL*.

(g) Scenario 3 / Training WL. (h) Scenario 3 / Test WL. (i) Scenario 3 / Test WL*.

Figure 6. Ratio of decisions in the unseen intersection scenario where the agent selected the actual fastest available vehicle (oracle action).

During training (Figure 5a), both MLP-DQN and PI-DQN exhibit relatively similar selection distributions, with both approaches concentrating their selections on lower-indexed actions (vehicles). However, the critical difference emerges during test deployment with unseen workloads (Figure 5b): MLP-DQN develops a pronounced positional bias, favoring vehicles at indices 0-3 while almost entirely neglecting vehicles beyond index 12. In contrast, PI-DQN maintains a more balanced distribution, spreading selections across a broader range of vehicles with notably less concentration on specific indices. When evaluating on Test WL* (Figure 5c), which features lower system load, MLP-DQN's positional bias becomes even more extreme, while PI-DQN continues to distribute selections more uniformly. This demonstrates that despite zero-padding and action masking, MLP-DQN learns correlations with positional encodings during training that fail to generalize to new workload patterns and worsen under reduced system load. PI-DQN's attention-based set processing architecture successfully eliminates these ordering artifacts, resulting in more robust generalization and fairer vehicle utilization across diverse distribution shifts between training and test conditions. Similar results are observed for Scenario 2 and 3.

Figure 6 quantifies agents' decision quality by measuring the fraction of times an agent correctly selected the best available resource, i.e., fastest processing vehicle (oracle action) across all three scenarios and workload conditions. The results reveal a strong difference in generalization capability: During training workload deployment (Figures 6a, 6d and 6g), MLP-DQN already exhibits suboptimal performance with success rates of 84.6 %, 69.0 % and 72.8 % in Scenarios 1, 2, and 3, respectively, indicating incomplete learning even on the training distribution. Meanwhile, PI-DQN achieves perfect 100 % oracle selection across all scenarios.

The performance gap degrades further under Test WL (Figures 6b, 6e and 6h), where MLP-DQN's success rate

decreases to 56.2 %, 40.0 %, and 42.7 % across the three scenarios. This severe degradation occurs despite the agent having been trained with action masking and zero-padding, confirming that the learned positional correlations fail when dealing with distribution shifts in task complexity and vehicle capabilities. PI-DQN maintains perfect 100 % accuracy across all test scenarios.

The difference in performance between the two approaches is most pronounced under Test WL* (Figures 6c, 6f and 6i). Here, MLP-DQN's decision quality deteriorates even further to 30.3 %, 18.3 %, and 18.6 %, representing an substantial performance decrease compared to training. This extreme degradation under reduced load occurs because: (i) with more idle vehicles, action masking imposes fewer constraints, allowing the agent to freely exploit its positional preferences, and (ii) infrequent task arrivals provide sparse feedback, giving insufficient signal to correct the learned position-based strategy. Again, PI-DQN demonstrates perfect oracle selection across all scenarios.

These results demonstrate that the permutation-invariant architecture produces a policy that is highly robust to: (i) unseen traffic patterns and varying VMC sizes, (ii) substantial shifts in task complexity and vehicle processing capabilities, and (iii) changes in system load and task arrival intensity. The consistent 100 % oracle selection by PI-DQN across all tested conditions indicates that once permutation invariance is properly achieved through the attention-based set processing.

VI. Conclusion

In this paper, we investigated the generalization capability of two DRL-based task-offloading schedulers in VEC environments. We demonstrated that conventional MLP-based DQN architectures fail to generalize when deployed under previously unseen configurations.

Extensive evaluation across three realistic vehicular scenarios and different workload regimes, showed that MLP-DQN's oracle action selection rate degrades dramatically, from approximately 70–85 % on training workloads to as low as 18–30 % under test conditions. By replacing the fixed-size MLP with a lightweight permutation-invariant transformer-based approach, we achieved 100% oracle selection across all tested scenarios. The attention-based set processing eliminates ordering artifacts, enabling the policy to generalize robustly to unseen vehicular densities, mobility patterns, and system loads without retraining.

In future work, we plan to extend this approach to joint task–vehicle assignment and decentralized multi-agent scheduling under incomplete state information and realistic communication constraints. We also aim to refine the computational model to better account for the real-world VMC dynamics, evaluate the computational overhead of the proposed scheduler, and further compare our approach with other state-of-the-art methods.

References

[1] C. E. Casetti, C. F. Chiasserini, F. Dressler, A. Memedi, D. Gasco, and E. M. Schiller, "AI/ML-based Services and Applications for 6G-Connected and Autonomous Vehicles," *Elsevier Computer Networks*, vol. 255, p. 110 854, Dec. 2024.

[2] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Towards Low-Latency Service Delivery in a Continuum of Virtual Resources: State-of-the-Art and Research Directions," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 4, pp. 2557–2589, 2021.

[3] L. Lin, X. Liao, H. Jin, and P. Li, "Computation Offloading Toward Edge Computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[4] F. Dressler, C. F. Chiasserini, F. H. P. Fitzek, H. Karl, R. Lo Cigno, A. Capone, C. E. Casetti, F. Malandrino, V. Mancuso, F. Klingler, and G. A. Rizzo, "V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing," *IEEE Network*, vol. 36, no. 3, pp. 24–31, May 2022.

[5] T. Higuchi, J. Joy, F. Dressler, M. Gerla, and O. Altintas, "On the Feasibility of Vehicular Micro Clouds," in *9th IEEE Vehicular Networking Conference (VNC 2017)*, Turin, Italy: IEEE, Nov. 2017, pp. 179–182.

[6] A. Boukerche and V. Soto, "Computation Offloading and Retrieval for Vehicular Edge Computing: Algorithms, Models, and Classification," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–35, Aug. 2020.

[7] M. Han, X. Sun, X. Wang, W. Zhan, and X. Chen, "Transformer-Based Distributed Task Offloading and Resource Management in Cloud-Edge Computing Networks," *IEEE Journal on Selected Areas in Communications*, vol. 43, no. 9, pp. 2938–2953, Sep. 2025.

[8] Y. Liu, H. Li, X. Vasilakos, R. Hussain, and D. Simeonidou, "Cooperative Task Offloading Through Asynchronous Deep Reinforcement Learning in Mobile Edge Computing for Future Networks," in *IEEE International Conference on Communications (ICC 2025)*, Montréal, Canada: IEEE, Jun. 2025, 1390â€"1395.

[9] Y. Ju, Y. Chen, Z. Cao, L. Liu, Q. Pei, M. Xiao, K. Ota, M. Dong, and V. C. M. Leung, "Joint Secure Offloading and Resource Allocation for Vehicular Edge Computing Network: A Multi-Agent Deep Reinforcement Learning Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5555–5569, May 2023.

[10] J. Shen, Y. Lin, Y. Zhang, W. Zhang, F. Shu, and J. Li, "Content Caching-Assisted Vehicular Edge Computing Using Multi-Agent Graph Attention Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 2, pp. 3509–3514, Feb. 2025.

[11] L. Geng, H. Zhao, J. Wang, A. Kaushik, S. Yuan, and W. Feng, "Deep-Reinforcement-Learning-Based Distributed Computation Offloading in Vehicular Edge Computing Networks," *IEEE Internet of Things Journal*, vol. 10, no. 14, pp. 12 416–12 433, Jul. 2023.

[12] K. Li, X. Wang, Q. He, M. Yang, M. Huang, and S. Dustdar, "Task Computation Offloading for Multi-Access Edge Computing via Attention Communication Deep Reinforcement Learning," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 2985–2999, Jul. 2023.

[13] A. Memedi, C. Lee, S. Ucar, O. Altintas, and F. Dressler, "Simulator for Reinforcement Learning-based Resource Management in Vehicular Edge Computing," in *16th IEEE Vehicular Networking Conference (VNC 2025), Poster Session*, Porto, Portugal: IEEE, Jun. 2025.

[14] Z. Zhou, A. Memedi, C. Lee, S. Ucar, O. Altintas, and F. Dressler, "Task Migration with Deadlines using Machine Learning-based Dwell Time Prediction in Vehicular Micro Clouds," *Elsevier High-Confidence Computing*, vol. 5, no. 2, p. 100 314, Jun. 2025.

[15] L. Johnston and R. W. Pazzi, "Keeping Information Alive: Hovering Information and Floating Content Paradigms for Vehicular Networks," in *International DCOSS*, Marina del Rey, CA: IEEE, May 2022, pp. 291–297.

[16] IEEE, "802.11bd-2022 - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 5: Enhancements for Next Generation V2X," IEEE, Std 802.11bd-2022, Mar. 2023.

[17] "3rd Generation Partnership Project;Technical Specification Group Services and System Aspects;Digital cellular telecommunications system (Phase 2+) (GSM);Universal Mobile Telecommunications System (UMTS);LTE;5G;Release description;Release 14," 3rd Generation Partnership Project, Sophia Antipolis, France, TS 21.914 v14.0.0, Jun. 2018.

[18] L. Codecá, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 52–63, Apr. 2017.