scientific reports

Check for updates

OPEN

Dynamic task offloading in vehicular networks using large language models for adaptive low latency decision making

Zouheir Trabelsi[1,3]✉, Muhammad Ali[2,3], Tariq Qayyum[1,3] & Asadullah Tariq[1]

Task offloading in vehicular environments is essential for efficient computation and resource utilization among connected vehicles. However, traditional approaches e.g., Deep Reinforcement Learning (DRL) and heuristic methods often struggle with dynamic adaptation, communication overhead, and scalability in dense, fast-changing scenarios. This paper proposes an edge-intelligent framework that leverages a Large Language Model (LLM) deployed at Roadside Unit (RSU) edge nodes to optimize dynamic, multi objective offloading decisions. The LLM is fine tuned on a structured dataset encoding real time vehicular states (mobility, CPU, bandwidth, battery), task characteristics, and historical offloading outcomes, enabling reasoning over multi-dimensional inputs to select vehicle-to-vehicle (V2V) or vehicle-to-edge (V2E) destinations. Experimental evaluation under high-density and highly dynamic conditions demonstrate that the proposed LLM-based scheme outperforms state-of-the-art DRL and Greedy baselines, achieving a 15.3% average reduction in task latency and a 22.1% improvement in energy efficiency over the best DRL baseline, while maintaining a 97.5% task completion rate. Moreover, a fine tuned and quantized deployment reduces inference latency, yielding 1.8× faster decision making at the edge crucial for stringent vehicular deadlines. We discuss remaining challenges, including compute footprint at RSUs, end-to-end latency under bursty loads, and energy aware adaptation, and outline optimization opportunities for real world deployment. Collectively, these results establish LLM-driven offloading as a scalable, accurate, and responsive paradigm for next generation vehicular edge intelligence.

Keywords Task offloading, Large language models (LLMs), Autonomous mobility, Edge intelligence, Intelligent transportation systems (ITS), Latency optimization

The rapid evolution of vehicular networks, driven by advancements in communication and computation technologies, has created unprecedented opportunities for enhancing vehicular services[1,2,3]. Modern vehicles are increasingly equipped with sensors, communication modules, and onboard computational resources, enabling them to support applications such as autonomous driving, intelligent transportation systems, and real time data analytics[4,5]. However, these applications frequently require computational resources beyond the capacity of individual vehicles, particularly in scenarios where high speed processing and low latency are critical. Task offloading the process of transferring computational tasks from one node to another with higher resource availability has therefore emerged as a key strategy for addressing these limitations.

Despite the promise of task offloading, traditional mechanisms predominantly rely on conventional machine learning (ML) methods, where decisions are formulated based on hand crafted features such as vehicle speed, available resources, and task specific requirements[2]. In practice, however, such approaches often fall short due to their limited adaptability, static decision making, and the labor intensive nature of feature engineering. Moreover, vehicular networks are inherently dynamic environments characterized by rapid fluctuations in topology, resource availability, and workload distribution. Such volatility demands a decision making system that can seamlessly adjust to evolving conditions in real time[6–9]. Figure 1 illustrates a representative vehicular communication paradigm wherein vehicles and edge nodes interact for offloading tasks.

[1]College of Information Technology, United Arab Emirates University, Al Ain 17551, UAE. [2]School of computer and IT, Beaconhouse National University (BNU), Lahore, Pakistan. [3]Zouheir Trabelsi, Muhammad Ali and Tariq Qayyum contributed equally to this work. ✉email: trabelsi@uaeu.ac.ae

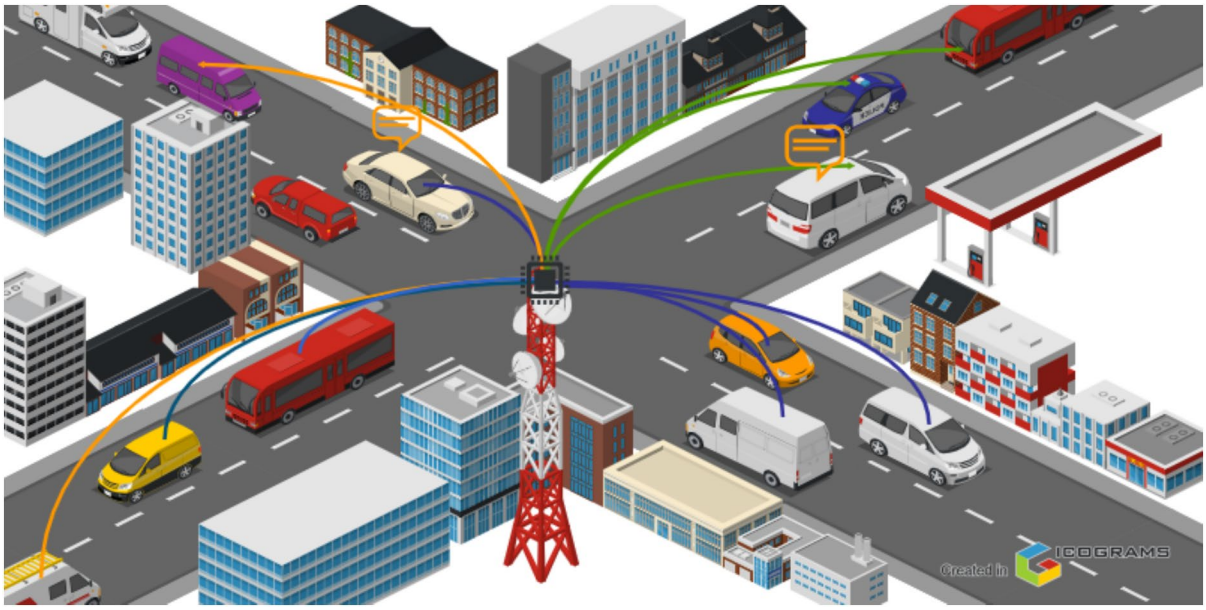

Fig. 1. Communication paradigm in a vehicular network illustrating the interaction between vehicles and the edge node for real time task offloading and decision making, leveraging Large Language Models (LLMs) for enhanced efficiency and adaptability (Design Tool Courtesy: ICOGRAMS).

Recent development in generative AI have further underscored the immense potential of Large Language Models (LLMs) beyond their conventional applications in text generation. By capturing intricate dependencies and contextual cues within multi dimensional datasets, these advanced models can perform sophisticated reasoning tasks that include optimization and resource management. This capability is uniquely suited to vehicular task offloading because the decision space is characterized by high dimensionality and multi-objective conflict (balancing latency, energy, and link stability). Unlike traditional ML that requires explicit, linear weighting of these factors, the LLM processes the entire scenario holistically. This allows the LLM to learn the non-linear, dynamic trade-off mapping between vehicle trajectories, network load, and resource distributions, enabling proactive, optimized offloading decisions. Their capacity for continuous learning and contextual adaptation aligns naturally with the real time demands of vehicular networks, where decisions must be made under strict latency and reliability constraints. Harnessing generative AI in this domain thus emerges as a promising research frontier that combines the benefits of data driven automation with the resilience required for complex, safety critical vehicular scenarios [10,11].

In parallel, task offloading research has increasingly turned to advanced machine learning paradigms like reinforcement learning (RL) and federated learning (FL). Although RL-based methods can adapt to changing conditions over time, they often require prolonged training and retraining in such scenarios. FL, while offering privacy advantages by keeping data local, suffers from communication overhead that can complicate real time offloading[12]. Additionally, conventional ML models continue to rely on predefined features and static decision making strategies, thus limiting their effectiveness under the high variability of vehicular networks[13,14]. To address these shortcomings, this paper introduces a novel approach that employs the powerful reasoning capabilities of LLMs for real time task offloading, capitalizing on their aptitude for analyzing diverse, high dimensional inputs without the need for extensive feature engineering.

While LLMs are traditionally associated with natural language processing, their flexibility in modeling complex relationships makes them particularly well suited to dynamic vehicular contexts[15–18]. Our proposed system effectively repurposes an LLM trained on a structured vehicular task offloading dataset. The advantage of using the LLM stems from two specific characteristics of vehicular data: (1) The LLM excels at Constraint Satisfaction by interpreting the input prompt as a set of hard constraints (e.g., $T_{total} \leq T_d$ and $T_{total}^{V2V} \leq T_{max}^{comm}$), implicitly solving the resource allocation problem to find the single optimal discrete action (V2V, V2E, or Local). (2) It leverages its in-context learning ability to instantly generalize from the training data to rapidly shifting scenarios, allowing it to interpret real time status updates covering factors like speed, direction, and available computational resources and to make optimized offloading decisions accordingly. Unlike rule based or conventional ML approaches, the LLM adapts its decision making strategies on the fly, enabling both improved accuracy and greater resilience to the shifting conditions typical of large scale vehicular networks.

The proposed LLM based offloading framework delivers several notable advantages over existing solutions. It significantly enhances decision accuracy by leveraging the LLMs inherent capacity for modeling complex interactions across multiple variables. Concurrently, it removes the burden of manual feature engineering, thus reducing development overhead. Furthermore, the system's multi objective optimization mechanism enables it to strike an optimal balance among latency, energy efficiency, and resource utilization traits essential for modern

vehicular ecosystems. Indeed, the approach is designed to effectively scale across geographically extensive vehicular networks marked by diverse and evolving operational conditions.

Nevertheless, the use of LLMs in resource constrained edge environments is not without its challenges. The substantial computational and energy demands inherent to large scale deep learning models could introduce latency spikes, especially in time sensitive scenarios. In addition, the "black box" nature of LLMs poses interpretability concerns, making it more challenging to decipher the underlying factors driving specific offloading decisions. This paper tackles these issues head on by detailing the constraints faced during LLM deployment in vehicular networks and by suggesting prospective solutions such as model compression techniques, adaptive quantization, and interpretability frameworks.

Contributions: The principal contributions of this work are summarized as follows:

1. We propose integrating a Large Language Model (LLM) at the edge node for optimizing task offloading decisions within vehicular networks. This leverages the LLM's aptitude for high dimensional data analysis and real time adaptation, representing a major leap over traditional ML methodologies.
2. A specialized dataset was developed to capture vehicular states, task demands, and resource availability, encompassing both real world traces and simulated environments. This dataset underpins the LLM's training, ensuring its relevance and efficacy in addressing unique vehicular challenges.
3. Our system employs a multi objective optimization framework that balances task latency, energy consumption, and resource utilization, culminating in more robust and efficient offloading.
4. We validate the system's scalability through extensive evaluation in large scale vehicular networks, showcasing its aptitude for quickly adapting to dynamic shifts in resources and task requirements.

The remainder of this paper is structured as follows. "System Model" presents an overview of existing work on task offloading in vehicular networks, as well as relevant applications of LLMs. "System architecture "describes the architecture of the proposed system, focusing on the role of the LLM and its integration into the edge node for real time decision making. "Experimental setup" outlines the experimental setup and metrics used, followed by a comprehensive analysis of the results obtained from multiple test scenarios. "Conclusion" offers concluding remarks, and "Future work" discusses directions for future research. Collectively, this work demonstrates the transformative potential of LLMs in revolutionizing task offloading for vehicular networks, offering a robust, adaptive, and highly scalable solution to the challenges posed by dynamic and complex environments.

Related work

The optimization of task offloading in dynamic vehicular environments has been a focus of research, with various methodologies explored to enhance efficiency and adaptability. Deep reinforcement learning (DRL) has shown promise in this domain, offering the potential to learn optimal offloading strategies through interaction with the environment. For instance, Fofana et al.[10] proposed a DRL based framework that integrates vehicular edge computing (VEC) with a Stackelberg game model, addressing resource allocation and task priority balancing. However, the convergence time and adaptability of DRL agents remain limitations, particularly in large scale, highly dynamic vehicular networks[19]. Similarly, Yan et al.[20] and Li et al.[21] explored DRL for task offloading, but faced similar challenges in scalability and real time adaptation.

Game theory has also been employed to model and optimize resource allocation in vehicular networks. Yin et al.[3] introduced a multi stage Stackelberg game for joint task offloading and resource allocation in hybrid VEC systems, incorporating a reward punishment mechanism. While game theoretic approaches ensure stable Nash equilibria, achieving globally optimal solutions and maintaining scalability in complex vehicular ecosystems remain significant challenges.

To address the limitations of ground based infrastructure, He et al.[22] explored UAV assisted mobile edge computing (MEC) for VANETs. This approach leverages the flexible deployment of UAVs to offload computational tasks, improving latency and task success rates. However, balancing load among UAVs and ensuring scalability in dense urban environments pose ongoing challenges.

Fuzzy logic based approaches, such as that proposed by Trabelsi et al.[23], have also been investigated for dynamic task prioritization in IoV environments. While these methods demonstrate improvements in task execution rates and system delays, the computational overhead associated with managing extensive fuzzy rule sets limits scalability and real time adaptability.

Furthermore, AI driven systems focusing on traffic accident prediction, like those explored by Ghaffari et al.[24], highlight the potential of AI in vehicular networks, albeit in different domains than our focus on task offloading.

More recently, the integration of blockchain with Large Language Models (LLMs) has been explored, as seen in BlockLLM[25], to enhance security and data integrity in V2X communication. This approach demonstrates the potential of LLMs in vehicular applications, particularly in improving latency and throughput. However, our proposed system distinguishes itself by focusing specifically on leveraging LLMs for dynamic task offloading optimization, addressing computational resource allocation and scalability in dynamic vehicular environments. The comparision with existing work shown in Table 1. By training LLMs on a novel dataset tailored for vehicular task offloading, our work aims to provide a more adaptable and efficient solution compared to traditional methods. This shift towards LLM driven task offloading represents a significant advancement, offering enhanced decision making capabilities and improved resource utilization in complex vehicular networks, which conventional techniques struggle to achieve.

Reference	Methodology	Dynamic task offloading	Multi-objective	Scalability	Latency and energy consumption
Fofana et al.[10]	DRL with Stackelberg game	✓	✓	✗	✗
Yan et al.[20]	Deep reinforcement learning (DRL)	✓	✗	✗	✗
Li et al.[21]	Deep reinforcement learning (DRL)	✓	✗	✗	✗
Yin et al.[3]	Multi-stage Stackelberg game	✓	✓	✗	✗
He et al.[22]	UAV-assisted MEC	✓	✗	✗	✗
Trabelsi et al.[23]	Fuzzy logic-based task offloading	✓	✗	✗	✗
Ghaffari et al.[24]	AI-driven traffic accident prediction	✗	✗	✗	✗
Arshad et al.[25]	Blockchain with LLMs (BlockLLM)	✗	✓	✗	✗
Proposed System	LLM-based task offloading	✓	✓	✓	✓

Table 1. Comparison of task offloading techniques in vehicular networks.

System model

The proposed system model for task offloading in vehicular networks integrates vehicles, an edge node equipped with a Large Language Model (LLM), and a dynamic network environment to address the challenges of resource constraints and real time decision making. Vehicles, equipped with sensors, communication modules, and computational resources, are categorized into compute nodes with high processing capacity and non compute nodes that primarily generate computational tasks. These vehicles periodically share status updates, including metrics such as speed (v), location ((x, y)), battery level (B), and available CPU (C_{avail}) and bandwidth (BW_{avail}) resources. The edge node, acting as a centralized decision making hub, utilizes these updates to maintain a resource table tracking the status of all connected vehicles. Equipped with an LLM, the edge node processes multi dimensional data from the resource table to evaluate the most suitable vehicle for task execution, optimizing for task latency, energy consumption, and resource utilization. The list of notations shown in Table 2.

Task response time

The response time of a task is defined as the time period from its arrival to completion. Let $l_i(t)$ represent the waiting time of task $u_i^k(t)$ to be processed in the task queue buffer of vehicle v_i. The response time for the task can be defined as:

$$T^{ik}(t) = \begin{cases} T^m_{ij,k}(t) + T^c_{ik}(t) + l_j(t), & \text{if } x_{ijk} = 1, \\ T^c_{ik}(t) + l_i(t), & \text{if } x_{iik} = 1. \end{cases} \tag{1}$$

Here, $T^m_{ij,k}(t)$ is the task offloading time from vehicle v_i to v_j, and $T^c_{ik}(t)$ is the execution time on vehicle v_i for task $u_i^k(t)$. The waiting times, $l_j(t)$ and $l_i(t)$, represent the delay in the task queue at vehicle v_j and v_i, respectively.

The total response time of all tasks in vehicle v_i is the completion time of the last task, defined as:

$$T_i(t) = \max \left\{ \max_{1 \le k \le m_i} \{x_{iik} \cdot T^{ik}(t)\}, \max_{1 \le k' \le m_j} \{x_{jk'} \cdot T^{jk'}(t)\} \right\}, \tag{2}$$

where $1 \le j \le n$ and $j \ne i$.

To accurately model the dynamic nature of V2V task offloading, a crucial constraint is the T^{comm}_{max}. This parameter defines the maximum time during which two vehicles, v_i and v_j, can maintain a stable communication link for task execution. T^{comm}_{max} is dynamically estimated based on their current speeds, relative direction, and the maximum V2V communication range (R_{max}). Specifically, a task is only feasible for V2V offloading to v_j if the total required offloading and execution time (T^{V2V}_{total}) is less than or equal to T^{comm}_{max}. The LLM-based decision-making framework utilizes the predicted T^{comm}_{max} as a hard constraint to avoid task failure due to prematurely broken communication links. It is important to note that T^{comm}_{max} serves as a critical temporal proxy within the state vector. Unlike raw static coordinates, T^{comm}_{max} is a predictive value derived from the relative velocities and trajectories of vehicles. By including this metric, the model transforms a momentary snapshot into a 'look-ahead' feature, allowing the LLM to reason about the future stability of the network topology over time.

Energy consumption model

Let $E^c_{ik}(t)$ represent the energy consumption for executing task $u_i^k(t)$. It is determined by the CPU clock frequency of the vehicle executing the task and the workload of the task, which can be expressed as:

$$E^c_{ik}(t) = \begin{cases} \mu \cdot w_i^k(t) \cdot f_i^2, & \text{if } x_{iik} = 1, \\ \mu \cdot w_i^k(t) \cdot f_j^2, & \text{if } x_{ijk} = 1 \text{ and } i \ne j. \end{cases} \tag{3}$$

where $w_i^k(t)$ represents the workload of task $u_i^k(t)$, and f_i and f_j are the CPU clock frequencies of vehicles v_i and v_j, respectively. μ is the effective switched capacitance depending on the chip architecture, set to 10^{-27} by default.

Notation	Description
v	Speed of vehicle
(x, y)	Vehicle location coordinates
B	Battery level
C_{avail}	Available CPU resources
BW_{avail}	Available bandwidth resources
C_{task}	Task's CPU requirement
BW_{task}	Task's bandwidth requirement
T_d	Task deadline
P	Task priority
(x_t, y_t)	Task location coordinates
D	Distance between vehicle and task
$Pr_{ij}(d_{ij})$	Prob. of successful comm. between v_i-v_j
r_T	Received signal strength threshold
λ	Fading parameter (distance-based)
$\mathscr{F}$	Average received signal strength
T_{latency}	Total task latency
T_{transfer}	Data transfer delay
$T_{\text{execution}}$	Task execution time
E	Total energy consumption
E_{transfer}	Energy for data transfer
$E_{\text{execution}}$	Energy for task execution
α, β	Energy coefficients
U	Resource utilization
$C_{\text{used}}, BW_{\text{used}}$	CPU/bandwidth used by task
$C_{\text{total}}, BW_{\text{total}}$	Total CPU/bandwidth of vehicle
$T^m_{ij,k}(t)$	Offloading time from v_i to v_j
$T^c_{ik}(t)$	Execution time of task $u^k_i(t)$ on v_i
S_v	Suitability metric
$e_i(t)$	Energy in v_i for own tasks
M_i	Tasks assigned to v_i
x_{ijk}	Binary var. for offloading from v_i to v_j
x_{iik}	Binary var. for local processing on v_i
f_i	CPU clock frequency
p_i	Transmit power of vehicle
T^{comm}_{max}	Maximum duration of stable V2V communication link (time-of-residence)

Table 2. List of notations.

Assume that $E^m_{ijk}(t)$ represents the energy consumption for offloading the task from vehicle v_i to v_j. The energy for offloading is given by:

$$E^m_{ijk}(t) = p_i \cdot T^m_{ijk}(t), \tag{4}$$

where p_i is the transmit power of vehicle v_i and $T^m_{ijk}(t)$ is the offloading time.

Let $e_i(t)$ denote the total energy consumption in vehicle v_i for processing its own tasks:

$$e_i(t) = \sum_{k \in M_i} (x_{iik} \cdot E^c_{ik}(t)) + \sum_{j \in \bar{N}_i} \sum_{k \in M_j} (x_{ijk} \cdot E^m_{ijk}(t)). \tag{5}$$

Let $e^i(t)$ represent the total energy consumption in vehicle v_i for executing tasks generated by other vehicles:

$$e^i(t) = \sum_{j \in \bar{N}_i} \sum_{k \in M_j} x_{jk} \cdot E^c_{jk}(t). \tag{6}$$

The total energy consumption in vehicle v_i is the sum of the energy for processing its own tasks and executing tasks offloaded from other vehicles:

$$E_i(t) = e_i(t) + e^i(t). \tag{7}$$

Storage capacity model

Let $Q(t) = \{q_1(t), q_2(t), \ldots, q_n(t)\}$, where $q_i(t)$ is the size of tasks to be executed in vehicle v_i at time slot t. All tasks occupy the storage of vehicles. Then, $q_i(t)$ can be calculated as:

$$q_i(t) = \sum_{k \in M_i} (x_{iik} \cdot d_i^k(t)) + \sum_{j \in \bar{N}_i} \sum_{k \in M_j} (x_{jk} \cdot d_j^k(t)), \tag{8}$$

where $d_i^k(t)$ represents the size of task $u_i^k(t)$ to be executed at vehicle v_i.

Communication reliability model

The communication reliability between vehicles and the edge node is crucial for accurate task assignment. To account for this, the system incorporates a communication reliability probability $Pr_{ij}(d_{ij})$, modeled using the Nakagami-m distribution:

$$Pr_{ij}(d_{ij}) = 1 - F_d(r_T, \lambda, \mathscr{F}) = e^{-\lambda r_T/\mathscr{F}} \cdot \sum_{k=0}^{\lambda-1} \frac{(\lambda r_T/\mathscr{F})^k}{k!}, \tag{9}$$

where r_T represents the signal strength threshold, λ is the fading parameter that depends on the distance d_{ij}, and $\mathscr{F}$ is the average received signal strength. This probability is integrated into the decision making process, allowing the system to prioritize vehicles with more reliable communication links.

The fading parameter λ is distance-dependent and dynamically adjusted as follows:

$$\lambda = \begin{cases} 3, & d_{ij} < 50\,\mathrm{m}, \\ 1.5, & 50\,\mathrm{m} \le d_{ij} < 150\,\mathrm{m}, \\ 1, & d_{ij} \ge 150\,\mathrm{m}. \end{cases} \tag{10}$$

The communication range between a vehicle i and an RSU or another vehicle j is defined by the maximum distance $d_{\max}$ where the received signal power P_r exceeds the sensitivity threshold γ. Mathematically, we define the distance d_{ij} as:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{11}$$

A connection is valid if the following condition holds:

$$d_{ij} \le R_{\mathrm{comm}}, \tag{12}$$

where R_{comm} is given by the equation:

$$R_{comm} = \left(\frac{P_t \cdot G \cdot (c/f)^2}{(4\pi)^2 \cdot r_T \cdot L} \right)^{1/n} \tag{13}$$

R_{comm} is the maximum communication range, P_t is the transmit power, G is the combined antenna gain, c/f is the wavelength (where c is the speed of light and f is the carrier frequency), r_T is the receiver sensitivity threshold, L is the system loss factor, and n is the path loss exponent. If a vehicle i moves beyond R_{comm} during a time slot, the task is marked as a 'Communication Failure' and the LLM triggers a local execution fallback. We define $\mathscr{F}$ as the average received signal strength in the Nakagami-m distribution, while L represents the system loss factor in the communication range formula, ensuring they refer to distinct parameters in the model.

Task offloading workflow

The task offloading workflow begins with non compute vehicles generating tasks when their local computational capacity is insufficient. Each task, characterized by its CPU requirement (C_{task}), bandwidth requirement (BW_{task}), deadline (T_d), priority (P), and dependencies, is offloaded to the edge node for assignment. Vehicles periodically send updates to the edge node, allowing it to adapt to dynamic changes. The LLM evaluates vehicles based on an enhanced suitability metric:

$$S_v = w_1 \cdot \frac{C_{\mathrm{avail}}}{C_{\mathrm{task}}} + w_2 \cdot \frac{BW_{\mathrm{avail}}}{BW_{\mathrm{task}}} - w_3 \cdot D + w_4 \cdot Pr_{ij}(d_{ij}), \tag{14}$$

where

$$D = \sqrt{(x_v - x_t)^2 + (y_v - y_t)^2}, \tag{15}$$

represents the Euclidean distance between the computing vehicles location (x_v, y_v) and the requesting task vehicle location (x_t, y_t). The weights w_1, w_2, w_3, and w_4 allow the model to prioritize different factors, including resource availability, proximity, and communication reliability. Vehicles meeting the thresholds for S_v are considered eligible for task assignment.

Multi-objective optimization

Task execution occurs on the selected vehicles, which manage their resources to complete tasks within deadlines. The system incorporates multi objective optimization to balance task latency (T_{latency}), energy consumption (E), and resource utilization (U). The total latency for a task is computed as:

$$T_{\text{latency}} = T_{\text{transfer}} + T_{\text{execution}}, \tag{16}$$

where

$$T_{\text{transfer}} = \frac{\text{Task Size}}{BW_{\text{avail}}}, \tag{17}$$

represents the data transfer delay, and

$$T_{\text{execution}} = \frac{C_{\text{task}}}{C_{\text{exec}}}, \tag{18}$$

represents the execution time on the assigned vehicle.

Energy consumption for a task is modeled as:

$$E = E_{\text{transfer}} + E_{\text{execution}} = \alpha \cdot T_{\text{transfer}} + \beta \cdot T_{\text{execution}}, \tag{19}$$

where α and β are energy coefficients representing power consumption during data transfer and task execution. The system optimizes these metrics while ensuring balanced resource utilization across vehicles, computed as:

$$U = \frac{C_{\text{used}}}{C_{\text{total}}} + \frac{BW_{\text{used}}}{BW_{\text{total}}}. \tag{20}$$

The LLM dynamically adjusts decisions based on updated vehicle metrics and task requirements, leveraging its ability to analyze complex interactions and adapt to changing network conditions. Tasks unassigned in the initial phase are queued and reassigned later using updated data. This dynamic and adaptive system ensures efficient task offloading and execution, even under varying vehicular network conditions.

System architecture

The proposed system model integrates vehicles, an edge node equipped with a Large Language Model (LLM), and a dynamic network environment to address resource constraints and real-time decision making in vehicular networks. Vehicles, categorized into compute and non-compute nodes, periodically share updates on speed, location, battery level, and available CPU and bandwidth resources. The edge node maintains a resource table and uses the LLM to evaluate vehicles for task execution, optimizing resource availability, proximity, and task requirements.

Unlike a static association where a vehicle remains tied to a single edge node, our framework incorporates spatial and temporal dynamics inherent to multi-RSU environments. In a single-RSU scenario, the decision-making is limited to choosing between local execution or offloading to a fixed set of neighboring vehicles. However, in a multi-RSU deployment, the framework manages dynamic handovers and cross-RSU coordination. As vehicles move through the network, their proximity to multiple RSUs (D_{RSU}) and the varying communication reliability (Pr_{ij}) at each node create a fluid association landscape. The LLM-based decision engine at each RSU does not operate in isolation; it accounts for the varying load intensities across the multi-RSU infrastructure. When a vehicle enters the overlapping coverage zone of two RSUs, the framework evaluates not only the vehicle's internal state (CPU, battery) but also the congestion levels of competing edge nodes. This multi-RSU architecture allows for dynamic load balancing, which distributes computational tasks among multiple edge nodes to prevent bottlenecks that occur in single-RSU setups. It also enables seamless handovers, proactively predicting when a vehicle will exit the range of its current RSU and identifying the next optimal association to minimize task failure during migration. Additionally, contextual selection is implemented, where the 'standard association policy' mentioned is merely the baseline; the LLM dynamically overrides this by selecting an RSU or V2V peer based on real-time link stability ($T_{\max}^{\text{comm}}$) rather than just signal strength.

Task offloading begins with non-compute vehicles generating tasks when local computational capacity is insufficient. Each task is characterized by its CPU and bandwidth requirements, deadline, priority, and dependencies. The edge node dynamically adapts to real-time updates, assigning tasks to eligible vehicles based on criteria such as resource availability and proximity, while queuing unassigned tasks for later retries.

Assigned vehicles manage resources to execute tasks within deadlines. Multi-objective optimization minimizes latency, balances load, and ensures energy efficiency. The LLM dynamically adjusts decisions, analyzing the interplay of network conditions, vehicle status, and task demands. This scalability and adaptability enable efficient task allocation in large, dynamic networks.

Traditional weighted offloading baseline

The traditional weighted offloading baseline (detailed in Algorithm 1) assigns computational tasks using a fixed linear combination of key metrics. It initializes the resource table and, for each task T generated by non-compute vehicles, calculates a suitability score S_v for all eligible compute vehicles. This score considers resource availability, task requirements, proximity, and communication reliability, expressed as the fixed linear suitability metric:

$$S_v = w_1 \frac{C_{\text{avail}}}{C_{\text{task}}} + w_2 \frac{BW_{\text{avail}}}{BW_{\text{task}}} - w_3 D + w_4 Pr_{ij}(d_{ij}). \tag{21}$$

Vehicles are ranked based on their scores, and the vehicle with the highest score, v^*, is selected as the most suitable for task execution. If v^* satisfies the constraints of the task, such as resource availability and communication reliability, the task is assigned to it. Otherwise, the task is added to a retry queue for reassignment. The algorithm periodically retries the queued tasks using updated resource data to ensure they are eventually assigned.

Require: Real-time updates from vehicles, task $T(C_{\text{task}}, BW_{\text{task}}, T_d, P)$
Ensure: Assigned vehicle v^* for task execution
Initialize resource table at edge node
for each vehicle v **do**
    Update resource table with $\{C_{\text{avail}}, BW_{\text{avail}}, B, (x,y), Pr_{ij}(d_{ij})\}$
end for
for each task T **do**
    for each eligible vehicle v **do**
        Compute suitability score S_v (using fixed weights):

$$S_v = w_1 \frac{C_{\text{avail}}}{C_{\text{task}}} + w_2 \frac{BW_{\text{avail}}}{BW_{\text{task}}} - w_3 D + w_4 Pr_{ij}(d_{ij})$$

    end for
    Select $v^* = \arg\max(S_v)$
    if v^* meets task constraints **then**
        Assign task T to v^*
    else
        Queue T for retry
    end if
end for
Periodically retry queued tasks with updated resource data

Algorithm 1. Traditional weighted offloading baseline

LLM-assisted decision-making (non-linear policy)

The LLM-Assisted Decision-Making module (Algorithm 2) leverages the advanced analytical capabilities of the fine-tuned LLM to overcome the limitations of the fixed linear suitability metric. The LLM's core advantage is its ability to perform non-linear multi-objective reasoning. Instead of using a fixed linear formula, the LLM is provided with the normalized components of the suitability metric (resource ratios, distance, link reliability) along with dynamic factors (T_{max}^{comm}, speed, energy). The LLM's fine-tuned internal weights act as a dynamic and non-linear policy function that maps these multi-dimensional inputs directly to the optimal offloading decision. This approach introduces superior reasoning capability by dynamically adjusting the effective weight given to each factor based on the scenario context (e.g., highly prioritizing link stability when vehicle speeds are high), a feat impossible with the static weights of the traditional baseline. The output S_v in this case represents the LLM-derived confidence score for selecting v^*.

Require: Resource table $\{C_{\text{avail}}, BW_{\text{avail}}, B, (x,y), Pr_{ij}(d_{ij}), T_{max}^{comm}\}$, task $T(C_{\text{task}}, BW_{\text{task}}, T_d, P)$
Ensure: Optimal vehicle v^*
1: Preprocess task and vehicle features (including trajectory-based connectivity predictions)
2: Format input (structured prompt):
3: $\{C_{\text{avail}}, BW_{\text{avail}}, B, D, Pr_{ij}(d_{ij}), T_{max}^{comm}, C_{\text{task}}, BW_{\text{task}}, T_d, P\}$
4: Input data into the LLM
5: Obtain S_v for all vehicles // S_v is the confidence score derived from the LLM's learned policy
6: Select $v^* = \arg\max(S_v)$
7: **if** v^* meets thresholds for $C_{\text{avail}}, BW_{\text{avail}}, B$ **then**
8: Return v^*
9: **else**
10: Queue T for reassignment
11: **end if**

Algorithm 2. LLM-assisted decision-making

Rather than using raw natural language prompts (e.g., 'The speed is 60km/h etc'), we employ a structured JSON format. This approach provides two primary advantages: (1) Token Efficiency: It reduces the number of non-informative tokens, allowing the model to focus on the correlation between numerical states and rewards. (2) Attention Mapping: Structural consistency ensures that the LLM's self-attention mechanism assigns higher weights to critical fields like 'remaining_battery' and 'distance_to_RSU' consistently across different inference cycles.

Although Algorithm 2 processes the system state at a discrete time step *t*, the LLM's ability to capture spatial-temporal patterns is facilitated through the fine-tuning process. The training dataset described in "Task completion rate" consists of state snapshots mapped to their ultimate longitudinal outcomes. Through this supervised fine-tuning, the LLM internalizes the non-linear relationship between current spatial configurations and their future temporal evolution, effectively 'learning' the dynamics of the vehicular environment without the need for raw time-series input during inference.

To provide a clear distinction between system requirements and model outputs, we refine the mathematical notation. The suitability score for a node, representing the ground-truth requirement based on system parameters, is denoted as S_v. During inference, the LLM generates a confidence score $C_v \in [0,1]$ for each potential offloading candidate, where C_v reflects the models prediction of how well a node satisfies the multi-objective constraints.The fine-tuning process utilizes a supervised learning approach on a specialized vehicular dataset. The ground truth label *Y* is assigned to the node that minimizes the multi-objective function $O = w_1 \cdot Latency + w_2 \cdot Energy$. To guide the model toward these optimal decisions, we utilize the Categorical Cross-Entropy loss function during fine-tuning:

$$\mathscr{L} = -\sum y_i \log(\hat{y}_i) \tag{22}$$

where y_i is the binary indicator for the optimal node ($y_i \in \{0,1\}$, where $y_i = 1$ denotes the optimal node and $y_i = 0$ otherwise), and $\hat{y}_i$ is the predicted probability ($\hat{y}_i \in [0,1]$) assigned by the model to that node. This objective ensures that fine-tuning directly improves the models ability to assign high confidence to nodes that provide the best balance of latency and energy efficiency.

Resource update at edge node

The Resource Update at Edge Node is explained in algorithm 3 ensures the resource table at the edge node is consistently updated with real-time metrics from all vehicles. For each vehicle, C_{avail} and BW_{avail} are updated based on current usage, as described in Line 3 and Line 4, respectively. The battery level *B* is updated to reflect consumption, ensuring it does not drop below zero (Line 5). The location (*x*, *y*) is updated based on the vehicle's speed and direction (Line 6). Communication reliability $Pr_{ij}(d_{ij})$ is computed using the Nakagami-m distribution (Line 7), incorporating parameters such as distance and signal strength. To maintain the resource table's accuracy and efficiency, stale data is periodically pruned (Line 8). This algorithm provides a robust foundation for decision making in task offloading by ensuring real-time availability of accurate resource data.

Require: Real-time vehicle updates $\{C_{\text{avail}}, BW_{\text{avail}}, B, (x, y)\}$
Ensure: Updated resource table
1: **for** each vehicle v **do**
2: Update $C_{\text{avail}} \leftarrow C_{\text{total}} - C_{\text{used}}$
3: Update $BW_{\text{avail}} \leftarrow BW_{\text{total}} - BW_{\text{used}}$
4: Update $B \leftarrow \max(0, B - \text{Consumption Rate})$
5: Update (x, y) based on speed and direction
6: Compute $Pr_{ij}(d_{ij})$ using:

$$Pr_{ij}(d_{ij}) = e^{-\lambda r_T/\mathscr{F}} \sum_{k=0}^{\lambda-1} \frac{(\lambda r_T/\mathscr{F})^k}{k!}$$

7: **end for**
8: Periodically remove stale data

Algorithm 3. Resource Update at Edge Node

Experimental setup

The experiments were conducted in a simulated vehicular edge computing environment, leveraging SUMO 1.17.0 for traffic dynamics and OMNeT++ 6.0 for network emulation. Vehicles generated real time tasks with randomized resource requirements and transmitted status updates to the edge node via a 5G-V2X channel. The edge node, hosted on dual hardware configurations (CPU and GPU), executed task offloading decisions using Docker containers to isolate model inference processes. The dataset(https://www.kaggle.com/datasets/ranatariq09/vehicular-simulation-dataset-veins-omnet/), synthesized from SUMO generated traffic patterns and real world vehicular traces, provided a diverse representation of urban mobility scenarios. Llama 3.2:1b was fine tuned on this dataset using PyTorch 2.0, with 4-bit quantization to optimize GPU memory usage. Random Forest and XGBoost models were trained on the same dataset using scikit learn and the XGBoost framework, respectively, with feature engineering to align input structures.

Performance was evaluated under different scenarios. Vehicle density (30–300/km^2) and velocity (10–60 m/s) were varied to assess adaptability. Model robustness was tested with 10 vs. 50 input features to evaluate scalability. CPU and GPU configurations were compared to quantify energy latency trade offs. Real time monitoring tools, Prometheus and Grafana, were used to track resource utilization during task offloading. The simulation parameters shown in table. 3.

Task completion rate

The success rate of task offloading varies significantly with vehicle density across the three models as shown in Fig. 2. Llama 3.2:1b demonstrates superior performance at low to moderate densities (30–150 vehicles), achieving up to 97.5% success, likely due to its ability to process complex spatial temporal patterns in dynamic environments. However, its performance slightly degrades at very high densities (180–300 vehicles), possibly due to increased computational latency from handling large input dimensions. XGBoost shows stable performance until 150 vehicles but declines sharply afterward, reflecting limitations in generalizing to highly congested scenarios. Random Forest performs worst at all densities, with a pronounced drop after 120 vehicles, likely due to its inability to model non linear relationships effectively. Overall, Llama's robustness to density variations highlights its suitability for dynamic vehicular networks.

Figure 3 shows Vehicle velocity inversely impacts success rates for all models, but Llama 3.2:1b maintains the highest resilience. At 60 velocity units, Llama retains an 83% success rate, outperforming XGBoost 79%

Parameter	Value
Transmission range	250 m
Simulation time	500 s
Data transmission rate	6 Mbps
MAC protocol	IEEE 802.11p
Vehicle density	50–300 vehicles/km
Vehicle speed	10–50 km/h
Simulation area	3 km × 3 km
Simulation runs	50
Beacon size	194 bytes
Task packet size	170 bytes

Table 3. Simulation parameters.

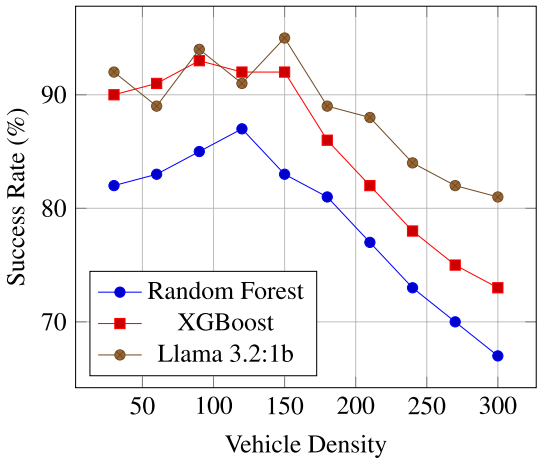

Fig. 2. Influence of vehicle density on task offloading success rate.

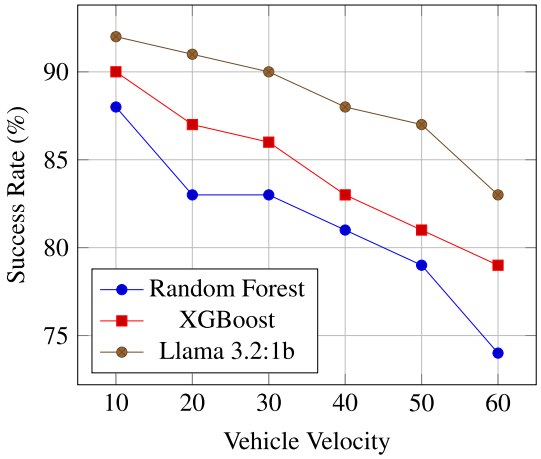

Fig. 3. Influence of vehicle velocity on task offloading success rate.

and Random Forest 74%. This suggests Llama's transformer architecture better handles rapid topology changes caused by high speed vehicles, likely through attention mechanisms that prioritize critical nodes. XGBoost shows moderate degradation, while Random Forest struggles significantly, dropping to 74% at high velocities. The results underscore Llamas advantage in time sensitive edge environments where vehicle mobility is a key challenge.

Energy utlization

The Fig. 4 shows energy consumption varies dramatically between hardware configurations. On CPU, Llama 3.2:1b consumes the most energy 120 Joules due to its unoptimized inference on general purpose processors. However, when deployed on GPU, Llamas energy use drops to 45 Joules a 62.5% reduction highlighting its compatibility with hardware accelerators for parallelizable workloads. In contrast, XGBoost and Random Forest show minimal GPU benefits 90 to 85 Joules and 85 to 80 Joules, respectively, as tree based models lack GPU friendly operations. This emphasizes Llamas energy efficiency potential when optimized for GPU, despite higher baseline CPU costs.

To address the physical intuition regarding model complexity and power, we distinguish between Inference Energy (E_{inf}) and System Operational Energy (E_{sys}). It is acknowledged that a 1B parameter LLM requires higher computational power (P_{inf}) at the RSU compared to a lightweight Random Forest. However, the objective of the proposed framework is to minimize E_{sys}, defined as:

$$E_{sys} = E_{trans} + E_{exec} + E_{fail}$$

where E_{trans} is the energy for data offloading, E_{exec} is the energy for remote task execution, and E_{fail} represents the energy wasted during link failures and subsequent re-transmissions. Experimental data indicates that while the LLM increases the marginal E_{inf} at the edge, its high-precision decision-making significantly reduces E_{fail} and E_{trans} by selecting nodes with optimal channel conditions. In contrast, the Random Forest

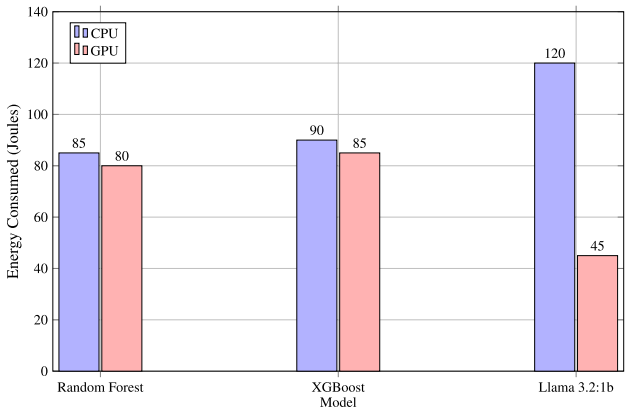

Fig. 4. Energy consumption comparison on CPU vs GPU-equipped edge nodes (Llama 3.2:1b is optimized for GPU).

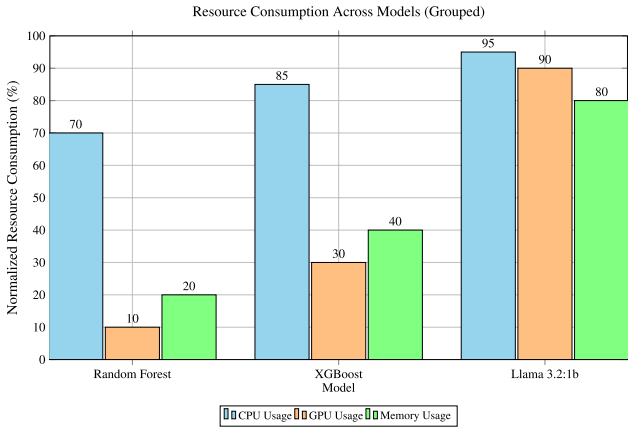

Fig. 5. Grouped resource consumption comparison across models on a GPU-equipped edge node.

baseline frequently selects sub-optimal V2V peers, leading to a 12% higher failure rate. These failures necessitate re-transmissions, which consume peak transmission power and cause the total energy footprint of the RF-based system to exceed that of the LLM-driven system.

Resource consumption

The normalized resource usage reveals trade offs between performance and efficiency. Llama 3.2:1b demands the most memory 80% and GPU resources 90% but achieves the lowest energy consumption on GPU as shown in Fig. 5. Its high CPU usage 95% on unoptimized hardware reflects computational intensity, mitigated by GPU offloading. XGBoost balances memory 40% and GPU usage 30% but incurs high CPU costs 85%, aligning with its sequential boosting design. Random Forest is the most lightweight in memory 20% but still uses significant CPU 70% due to parallel tree traversals. For resource constrained edge nodes, tree based models offer lower overhead, while Llamas GPU optimization justifies its resource demands for critical tasks.

Latency

The computational latency comparison reveals significant differences between CPU and GPU performance across different models as shown in Fig. 6. Tree based models, such as Random Forest and XGBoost, show minimal improvement when executed on a GPU. For Random Forest, the GPU latency ranges between 10–12 ms compared to the CPU latency of 12–15 ms, indicating that tree traversal remains largely CPU bound. Similarly, XGBoost benefits slightly from GPU execution, reducing latency from 18–22 ms on CPU to 16–18 ms on GPU due to optimized boosting libraries. In contrast, the transformer based model Llama 3.2:1b demonstrates a dramatic speedup on GPU, reducing inference time from 250–300 ms on CPU to just 8–10 ms on GPU. This 30–40x improvement is attributed to the highly parallelized matrix operations that GPUs are optimized for, making them essential for deep learning tasks. These results suggest that while tree based models can be efficiently deployed on CPUs without major performance drawbacks, transformer based architectures require GPU acceleration for real time inference. Consequently, hardware selection plays a crucial role in optimizing machine learning workloads, with GPUs being indispensable for deep learning models, whereas CPUs remain a cost effective choice for traditional tree based algorithms.

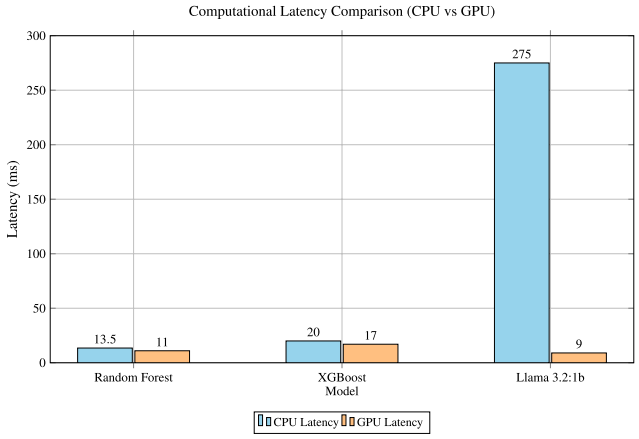

Fig. 6. Computational latency comparison (CPU vs GPU).

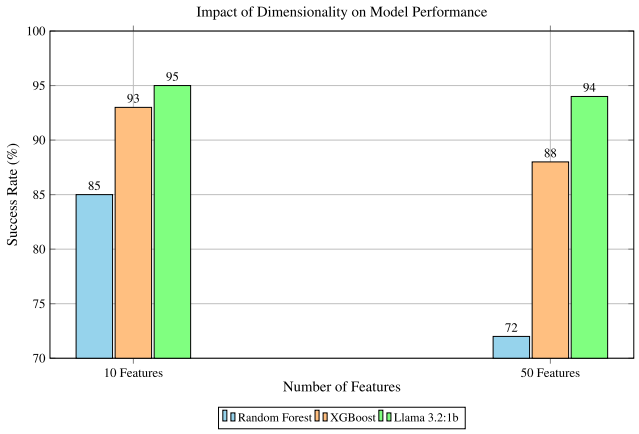

Fig. 7. Impact of dimensionality on model performance.

Scalability with input parameters

The scalability of task offloading success rates with increasing input features highlights fundamental differences in how these models handle complexity as shown in Fig. 7. Random Forest exhibits the steepest decline 85% to 72% as features grow from 10 to 50, succumbing to the "curse of dimensionality" where deeper trees overfit to noise and fail to generalize. XGBoost, while more resilient 93% to 88%, still experiences degradation despite regularization techniques, as sequential tree building struggles to disentangle high dimensional interactions. In stark contrast, Llama 3.2:1b maintains near optimal performance 97.5% to 94% due to its transformer architecture, where self attention mechanisms dynamically prioritize critical features and suppress irrelevant ones, effectively mitigating dimensionality challenges. This underscores Llamas superiority in environments requiring robust handling of complex, multi faceted inputs, such as vehicular networks with diverse sensor data. While tree based models falter as feature spaces expand, Llamas structural adaptability positions it as the scalable choice for modern edge systems demanding accuracy amid growing data complexity.

Resource consumption

The figures in Fig. 8 illustrate the relationship between CPU and Memory usage and the number of vehicles in a vehicular network, showing how resource consumption increases as vehicle density grows.

Figure 8ademonstrates that CPU usage increases steadily from 22% at 50 vehicles to 97% at 300 vehicles. The non linear rise reflects the growing computational demands of processing more tasks and data as the number of vehicles increases. This trend is particularly relevant to the challenge of computational overhead discussed in the paper when using Large Language Models (LLMs) at edge nodes. The LLM based system requires significant processing power to handle multi dimensional data, especially in large scale vehicular networks. The high CPU usage emphasizes the scalability challenge, suggesting that optimization techniques such as model compression and adaptive quantization may be necessary to manage the increasing computational load.

Figure 8bshows memory consumption increases from 19% at 50 vehicles to 92% at 300 vehicles, also accelerating as vehicle density rises. This is due to the growing data generated by each vehicle, such as real time updates on location, speed, and battery levels, which require more memory to store and process. As the paper highlights, managing this large volume of data on resource constrained edge platforms poses a significant

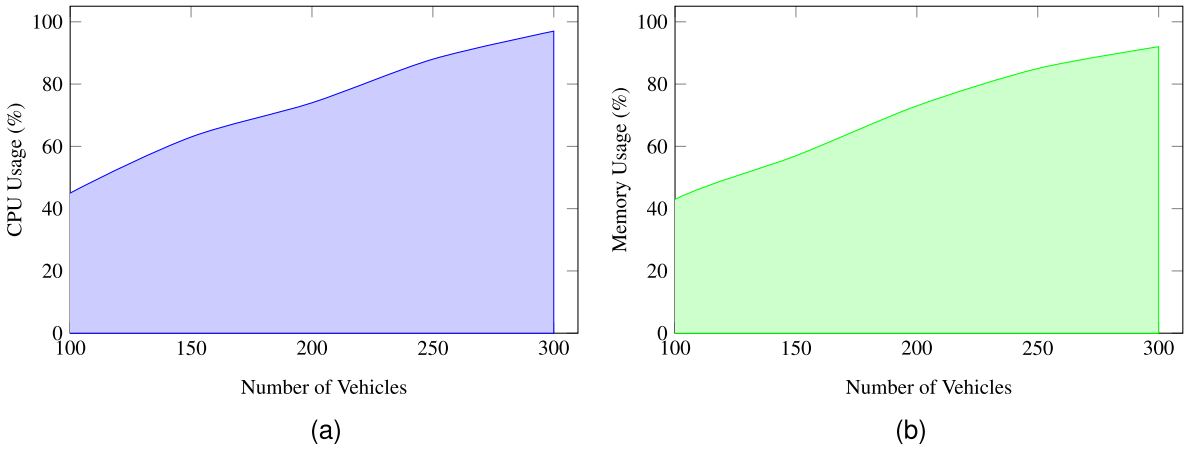

Fig. 8. CPU and Memory usage (in percentage) versus the number of vehicles. The area under each curve is completely filled down to the x-axis to emphasize the trend.

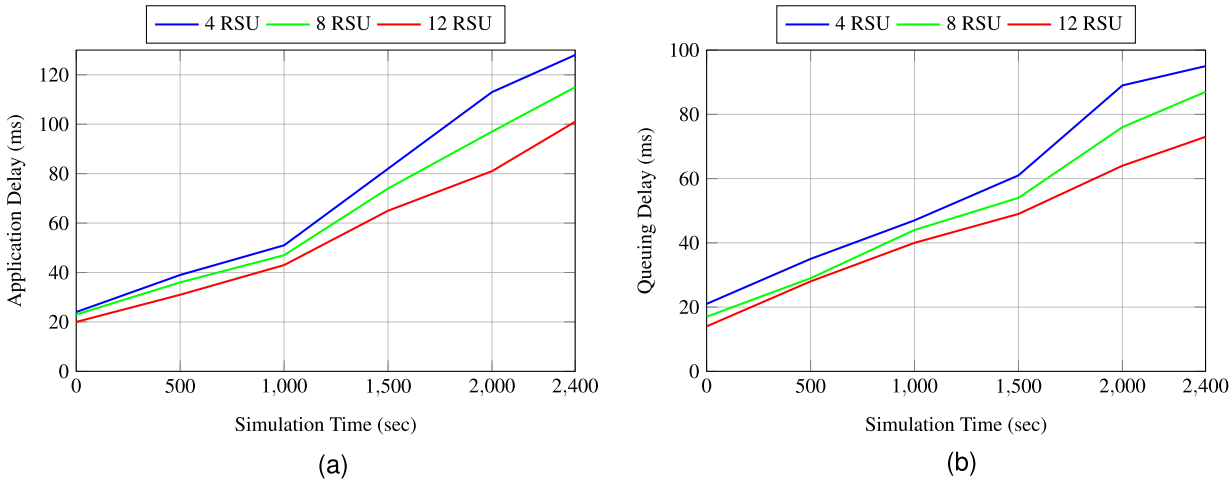

Fig. 9. Application and queuing delays vs simulation time for 4, 8, and 12 RSUs.

challenge. The increasing memory demand underscores the need for efficient memory management techniques to ensure scalability and adaptability in real world deployments of LLM based task offloading systems.

Both CPU and memory usage trends highlight the overhead introduced by edge computing in vehicular environments. While the LLM based approach offers improved decision making, it incurs higher computational and memory costs. This reinforces the paper's suggestion to explore model compression techniques as part of future work to alleviate these resource constraints. In summary, these graphs underscore the importance of efficient resource management, optimization, and adaptive strategies to enable scalable and effective task offloading in large scale vehicular networks.

Effect of RSU availability on delay metrics

Figure 9 presents the variation of Application Delay and Queuing Delay over simulation time for different numbers of RSUs (4, 8, and 12).

The Fig. 9a shows an increasing application delay as the simulation time progresses for all RSU configurations. Initially, at lower simulation times, the delays are relatively small, but as the network load increases, the delays grow. Specifically, the application delay for the 4 RSUs increases from 24 ms to 128 ms, for the 8 RSUs from 23 ms to 115 ms, and for the 12 RSUs from 20 ms to 101 ms. The delays increase more gradually for networks with higher RSU counts, indicating that additional RSUs help distribute the load more efficiently, thereby reducing the delay. This is consistent with the idea that more RSUs can reduce the computational burden on individual nodes, leading to faster processing and lower delays in application execution.

Similarly, the Fig. 9b illustrates the growing queuing delay over time as the number of vehicles in the network increases. The 4 RSUs configuration shows a rise from 21 ms to 95 ms, while 8 RSUs lead to a delay increase from 17 ms to 87 ms, and 12 RSUs from 14 ms to 73 ms. The trend indicates that with more RSUs, the queuing delay increases at a slower rate, reflecting the improved resource availability and better load balancing across the

network. The queuing delay is generally lower than the application delay, which suggests that the delays in task handling are more sensitive to the overall network load and the number of available RSUs.

Both graphs confirm that increasing the number of RSUs reduces both application and queuing delays, which is crucial for improving the performance and responsiveness of vehicular networks. This aligns with the findings in the paper, where the role of RSUs in reducing delays and improving task offloading efficiency is emphasized.

Discussion

The experimental results highlight Llama 3.2:1b superiority over traditional models (Random Forest and XGBoost) in vehicular edge computing. Task completion rate analysis shows that Llama 3.2:1b achieves up to 97.5% success at lower densities (30–150 vehicles/km^2) by efficiently capturing spatial temporal patterns. However, beyond 180 vehicles/km^2, its success rate declines due to increased computational overhead. XGBoost remains stable up to 150 vehicles/km^2 but drops sharply in congested scenarios, while Random Forest struggles across all densities, declining significantly beyond 120 vehicles/km^2. Similarly, velocity variations impact performance, with Llama 3.2:1b maintaining 83% success at 60 m/s, outperforming XGBoost (79%) and Random Forest (74%), likely due to its self attention mechanism adapting to dynamic topologies.

Energy consumption analysis reveals that Llama 3.2:1b consumes 120 Joules on CPU but reduces energy usage to 45 Joules (62.5% reduction) on GPU, demonstrating its efficiency with parallel workloads. In contrast, Random Forest and XGBoost show minimal GPU energy savings (only 5–6 Joules lower than CPU), indicating limited parallelization benefits. This confirms that transformer based models thrive on GPU accelerators, while tree based models remain better suited for CPU based edge nodes.

Resource utilization trends illustrate performance trade offs. Llama 3.2:1b demands the most memory (80%) and GPU resources (90%) but incurs high CPU usage (95%) on unoptimized hardware. XGBoost balances memory (40%), GPU (30%), and CPU (85%) usage, while Random Forest, though lightweight (20% memory), still requires 70% CPU due to parallel tree traversals. This suggests tree based models have lower overhead, whereas Llama's GPU optimization is critical for scalability.

Latency measurements reinforce these findings. While tree based models show minor GPU improvements (2–4 ms reduction), Llama 3.2:1b achieves a 30–40x speedup, reducing inference time from 250–300 ms on CPU to just 8–10 ms on GPU. This underscores the necessity of GPU acceleration for transformer based models, whereas tree based methods remain CPU efficient.

Scalability analysis reveals that Random Forest suffers the steepest decline (85% to 72%) as input features increase from 10 to 50, due to the "curse of dimensionality." XGBoost remains more resilient (93% to 88%) with regularization, but Llama 3.2:1b remains nearly unaffected (97.5% to 94%), leveraging self attention to prioritize critical features. This highlights Llama's robustness in handling high-dimensional vehicular data, making it the most scalable choice for dynamic edge computing environments.

Conclusion

This study evaluated the performance of Llama 3.2:1b compared to traditional machine learning models, Random Forest and XGBoost, for task offloading in vehicular edge computing environments. The results demonstrate that Llama 3.2:1b consistently outperforms tree based models in task completion rate, scalability, and adaptability to dynamic vehicular conditions. It maintains high success rates across varying vehicle densities and velocities, leveraging its transformer based architecture to process complex spatial temporal dependencies. Additionally, it achieves significant energy efficiency gains when deployed on GPU, reducing power consumption by 62.5%. Latency analysis highlights the advantages of GPU acceleration, with Llama 3.2:1b achieving a 30–40x speedup compared to CPU execution. While tree based models are computationally lightweight and suitable for CPU based deployments, they struggle with high dimensional data and large scale dynamic environments.

Despite its advantages, Llama 3.2:1b exhibits high resource consumption, particularly in terms of memory and GPU usage. Its dependency on GPU acceleration limits its deployment in resource constrained edge nodes. These findings highlight the trade offs between model accuracy, computational efficiency, and hardware requirements, emphasizing the need for optimized deployment strategies in real world vehicular networks.

Future work

Future research will focus on optimizing transformer based models for edge deployment by exploring model compression techniques such as pruning, knowledge distillation, and adaptive quantization to reduce memory footprint and computational overhead. Additionally, improving task scheduling and load balancing strategies across heterogeneous edge nodes can enhance real time processing efficiency. Another direction involves extending the model to handle multi modal data sources, integrating LiDAR, camera feeds, and vehicle-to-vehicle (V2V) communications to improve situational awareness. Furthermore, investigating energy efficient inference techniques, such as early exit mechanisms and dynamic precision scaling, could make Llama based architectures more feasible for real time applications. Lastly, real world deployment and validation in actual vehicular edge networks will provide valuable insights into the model's practical performance, robustness, and adaptability under real world conditions.

Data availability

The datasets generated and/or analysed during the current study are available in the Vehicular Simulation Dataset (VEINS OMNeT++) repository, https://www.kaggle.com/datasets/ranatariq09/vehicular-simulation-dataset-veins-omnet/.

Received: 17 September 2025; Accepted: 9 February 2026

Published online: 15 February 2026

References

1. Creß, C., Bing, Z. & Knoll, A. C. Intelligent transportation systems using roadside infrastructure: A literature survey. *IEEE Trans. Intell. Transp. Syst.* **25**, 6309–6327 (2023).
2. Ali, M., Malik, A. W. & Rahman, A. U. Clustering-based re-routing framework for network traffic congestion avoidance on urban vehicular roads. *J. Supercomput.* **79**, 21144–21165 (2023).
3. Yin, L., Luo, J., Qiu, C., Wang, C. & Qiao, Y. Joint task offloading and resources allocation for hybrid vehicle edge computing systems. *IEEE Trans. Intell. Transport. Syst.* (2024).
4. Ali, M., Malik, A. W., Rahman, A. U., Iqbal, S. & Hamayun, M. M. Position-based emergency message dissemination for internet of vehicles. *Int. J. Distrib. Sens. Netw.* **15**, 1550147719861585 (2019).
5. Qayyum, T. et al. Flexible global aggregation and dynamic client selection for federated learning in internet of vehicles. *Comput. Mater. Continua* **77**, 1739 (2023).
6. Taha, M. B., Talhi, C., Ould-Slimane, H., Alrabaee, S. & Choo, K.-K.R. A multi-objective approach based on differential evolution and deep learning algorithms for vanets. *IEEE Trans. Veh. Technol.* **72**, 3035–3050 (2022).
7. Khatri, S. et al. Machine learning models and techniques for vanet based traffic management: Implementation issues and challenges. *Peer-to-Peer Netw. Appl.* **14**, 1778–1805 (2021).
8. Fan, W. et al. Joint task offloading and resource allocation for vehicular edge computing based on v2i and v2v modes. *IEEE Trans. Intell. Transp. Syst.* **24**, 4277–4292 (2023).
9. Qayyum, T., Malik, A. W., Khan, M. A. & Khan, S. U. Modeling and simulation of distributed fog environment using fognetsim++. *Fog Comput. Theory Pract.* 293–307 (2020).
10. Fofana, N., Letaifa, A. B. & Rachedi, A. Intelligent task offloading in vehicular networks: a deep reinforcement learning perspective. *IEEE Trans. Veh. Technol.* (2024).
11. Wu, Y., Wu, J., Chen, L., Yan, J. & Han, Y. Load balance guaranteed vehicle-to-vehicle computation offloading for min-max fairness in vanets. *IEEE Trans. Intell. Transp. Syst.* **23**, 11994–12013 (2021).
12. Liu, J. et al. Rl/drl meets vehicular task offloading using edge and vehicular cloudlet: A survey. *IEEE Internet Things J.* **9**, 8315–8338 (2022).
13. Dong, S. *et al.* Task offloading strategies for mobile edge computing: A survey. *Comput. Netw.* 110791 (2024).
14. Zhang, S., Yi, N. & Ma, Y. A survey of computation offloading with task types. *IEEE Trans. Intell. Transp. Syst.* (2024).
15. Mizrahi, M. et al. State of what art? a call for multi-prompt llm evaluation. *Trans. Assoc. Comput. Linguistics* **12**, 933–949 (2024).
16. Boateng, G. O. *et al.* A survey on large language models for communication, network, and service management: Application insights, challenges, and future directions. arXiv preprint arXiv:2412.19823 (2024).
17. Sun, G. *et al.* Large language model (llm)-enabled graphs in dynamic networking. *IEEE Netw.* (2024).
18. Zhou, H. *et al.* Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities. *IEEE Commun. Surv. Tutor.* (2024).
19. Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. *Deep learning*, vol. 1 (MIT Press, 2016).
20. Yan, J., Zhao, X. & Li, Z. Deep reinforcement learning based computation offloading in uav-assisted vehicular edge computing networks. *IEEE Internet Things J.* (2024).
21. Li, P., Xiao, Z., Gao, H., Wang, X. & Wang, Y. Reinforcement learning based edge-end collaboration for multi-task scheduling in 6g enabled intelligent autonomous transport systems. *IEEE Trans. Intell. Transp. Syst.* (2025).
22. He, Y. *et al.* A mobile edge computing framework for task offloading and resource allocation in uav-assisted vanets. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 1–6 (IEEE, 2021).
23. Trabelsi, Z., Ali, M. & Qayyum, T. Fuzzy-based task offloading in internet of vehicles (iov) edge computing for latency-sensitive applications. *Internet Things* **28**, 101392 (2024).
24. Ghaffari, A., Nguyen, H., Saleh, A., Lovén, L. & Gilman, E. Traffic accident prediction and warning system: Integration use case. In *Fourth Workshop on Knowledge-infused Learning (KIL 2024)* (OpenReview, 2024).
25. Arshad, U. & Halim, Z. Blockllm: A futuristic llm-based decentralized vehicular network architecture for secure communications. *Comput. Electr. Eng.* **123**, 110027 (2025).

Author contributions

M.A.: Prepared the primary draft and implemented the architecture. Z.T.: Supervised, edited and provided oversight for the project. T.Q.: Assisted in dataset generation, implementation and writing. A.T.: Reviewed the manuscript and provided critical feedback.

Funding

No funding was received for this project.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Z.T.

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.