Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Proactive MEC Resource Allocation Using Deep Learning for Real-Time Collision Avoidance in Connected Vehicular Networks

Khadidja Zairi[1],Younes Guellouma[1],Bouziane Brik[2],and Hadda Cherroun[1]
[1]*LIM Lab, Amar Telidji University, Laghouat 03000, Algeria*
[2] College of Computing and Informatics, University of Sharjah, Sharjah, UAE

ARTICLE INFO

Received: 30 Dec 2024

Revised: 05 Feb 2025

Accepted: 25 Feb 2025

ABSTRACT

This paper proposes a proactive resource allocation framework for multi-access edge computing (MEC) systems in connected vehicular networks, targeting real-time collision avoidance applications. In such environments, dynamic workload variations caused by vehicle mobility and traffic density can lead to inefficient resource utilization and latency violations under conventional reactive allocation strategies. To address this challenge, we develop a deep learning-based prediction model using a Gated Recurrent Unit (GRU) to forecast short-term CPU and memory demand. The predicted workload is integrated into a latency-aware orchestration mechanism that performs proactive resource allocation under capacity constraints. Extensive simulations using real-world vehicular mobility traces demonstrate that the proposed approach significantly improves system performance. In particular, it reduces latency and SLA violations by more than 50% compared to reactive allocation strategies, while improving CPU utilization efficiency. The results highlight the effectiveness of combining deep learning-based workload prediction with proactive resource orchestration for next-generation MEC-enabled vehicular systems.

Keywords: Mobile Edge Computing (MEC), Resource Allocation, Deep Learning, GRU, Workload Prediction, Proactive Resource Management, SLA Violations, Vehicular Networks.

INTRODUCTION

The rapid proliferation of Internet of Things (IoT) devices and the emergence of real-time applications such as autonomous vehicles, augmented reality (AR), and virtual reality (VR) have imposed unprecedented demands on traditional cloud computing architectures. These systems struggle to meet strict latency requirements due to inherent backhaul delays and centralized processing limitations. Mobile Edge Computing (MEC) has therefore emerged as a key enabling paradigm that extends cloud capabilities to the network edge, providing low latency, improved Quality of Service (QoS), and localized data processing close to end users [1–3]. By deploying computational resources at the edge, MEC significantly reduces round-trip delay and alleviates core network congestion, making it particularly suitable for latency-sensitive applications in 5G and emerging 6G networks [4].

Despite these advantages, efficient resource allocation in MEC remains a critical challenge. The highly dynamic nature of workload demand, coupled with user mobility and limited server capacities, makes traditional resource management strategies insufficient. In particular, reactive allocation approaches provision resources only after task arrival, which often results in temporary overload, queue buildup, increased latency, and potential Service Level Agreement (SLA) violations [5]. These limitations highlight the need for intelligent, proactive resource management strategies that can adapt to time-varying workload conditions.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

In this context, machine learning (ML), particularly deep learning (DL), has demonstrated strong potential for modeling complex, non-stationary workload behaviors. By exploiting temporal patterns in historical resource usage data, DL models can accurately forecast future demand and enable anticipatory resource provisioning [6–8]. However, most existing approaches focus on isolated optimization problems, such as task offloading or single-resource allocation, while neglecting coordinated decision-making across multiple MEC servers. As a result, they fail to fully exploit the benefits of centralized orchestration in distributed edge environments.

To address these challenges, a unified proactive framework is required to jointly integrate workload prediction and resource orchestration. In this paper, we propose a deep learning-driven proactive resource allocation framework for multi-server MEC systems. The proposed approach leverages a Gated Recurrent Unit (GRU)-based model to predict short-term CPU and memory demand, which is then used by a centralized Edge Orchestrator to perform proactive resource allocation under capacity constraints. This enables anticipatory provisioning, reduces latency, and minimizes SLA violations. Extensive simulations using real vehicular mobility traces demonstrate that the proposed approach significantly outperforms conventional reactive strategies under dynamic workload conditions, achieving improved latency performance and higher resource utilization efficiency.

The primary objectives of our work are to design a DL-based workload prediction module for MEC environments, develop a proactive resource allocation strategy driven by predicted demand, reduce average task latency and SLA violations, and improve CPU utilization efficiency under bursty traffic conditions. The main contributions of this work are a unified proactive resource orchestration framework for multi-server MEC systems, a GRU-based multi-resource (CPU and memory) workload prediction model, an SLA-aware latency-driven resource allocation mechanism, and a comprehensive evaluation using real vehicular mobility traces.

The remainder of this paper is organized as follows. Section II reviews related work. Section III presents the proposed methodology and system model. Section IV describes the experimental setup and performance evaluation. Section V discusses the results, and Section VI concludes the paper.

RELATED WORK

Resource allocation in Mobile Edge Computing (MEC) has attracted significant research attention due to the stringent latency requirements of emerging 5G and beyond applications. Early approaches mainly relied on optimization-based techniques to address task scheduling and resource provisioning problems. While these methods provide theoretical guarantees, they often depend on static assumptions and struggle to adapt to highly dynamic workload variations and user mobility [7].

With the increasing complexity of MEC environments, machine learning (ML) and deep learning (DL) techniques have been widely adopted to enable intelligent and adaptive resource management. Djigal et al. [7] present a comprehensive survey of ML and DL approaches to MEC resource allocation, highlighting their advantages in handling dynamic, high-dimensional environments. However, the study also emphasizes remaining challenges in real-time scalability and coordination across multiple edge servers. Similarly, Ismail et al. [1] review deep reinforcement learning (DRL)-based scheduling strategies and identify key limitations, including high convergence complexity and the lack of global orchestration mechanisms.

Recent research has explored various learning-based approaches for improving resource allocation. For instance, autoencoder-enhanced DRL has been applied to large-scale online scheduling [15]. At the same time, graph-based methods, such as Graph Neural Networks (GNNs), have been used to model network topology to support more informed decision-making [16]. In addition, Cybertwin-driven deep learning frameworks have been proposed to optimize energy efficiency in MEC systems [19]. In the context of vehicular networks, Brik et al. [18] combined virtualization with LSTM-based mobility prediction to improve MEC resource dimensioning for collision avoidance applications. Complementary studies have also investigated workload prediction using CNN-, LSTM-, and Transformer-based models in cloud environments [17].

A growing body of work focuses on predictive resource allocation strategies. Zheng et al. [8] propose a dynamic multi-time-scale framework based on Lyapunov optimization for user admission and resource allocation in MEC systems. Although effective in handling stochastic task arrivals, their approach does not incorporate explicit workload

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

prediction. Similarly, Ye et al. [9] investigate joint deep neural network partitioning and resource allocation to minimize end-to-end delay. However, their work primarily targets inference optimization rather than proactive orchestration across multiple MEC servers.

Workload prediction using deep learning has also been explored to support proactive provisioning. For example, a recent study in healthcare-oriented MEC environments [10] employs a GRU-based model to forecast workload demand and reduce SLA violations. Despite its effectiveness, the approach is application-specific and does not consider coordination across multiple edge servers. Yu et al. [11] propose a deep learning-based task offloading and resource allocation strategy that improves system efficiency. Still, their solution focuses on local decision-making at individual edge nodes and lacks centralized orchestration.

Other studies have addressed resource allocation in heterogeneous MEC environments. Xu et al. [14] propose an energy-efficient task offloading and resource allocation scheme for NOMA-based networks that jointly optimizes subchannel allocation, power control, workload offloading, and CPU frequency. Although the approach effectively reduces energy consumption, it relies on complex optimization techniques and does not consider predictive or proactive mechanisms.

Deep reinforcement learning (DRL) has also been widely used for joint task offloading and resource allocation. For example, recent work in edge-cloud networks [12] leverages DRL to optimize decision-making under dynamic conditions, while reinforcement learning-based approaches have been applied in Internet of Vehicles (IoV) scenarios to reduce latency and energy consumption [2]. However, these methods typically rely on reactive or online learning strategies and do not explicitly integrate workload prediction for proactive resource allocation.

In summary, existing research demonstrates the effectiveness of ML, DL, and DRL techniques for MEC resource allocation. However, several important limitations remain. First, most approaches do not tightly integrate workload prediction with resource allocation decisions. Second, centralized orchestration across multiple MEC servers is often neglected. Third, limited attention has been given to latency- and SLA-aware proactive optimization. To address these limitations, this paper proposes a deep learning-driven proactive resource orchestration framework in which a centralized Edge Orchestrator predicts future workload and performs anticipatory resource allocation across multiple MEC servers. Unlike existing reactive or single-node approaches, the proposed method integrates prediction, orchestration, and SLA-aware latency optimization within a unified architecture.

METHODOLOGY

System Architecture

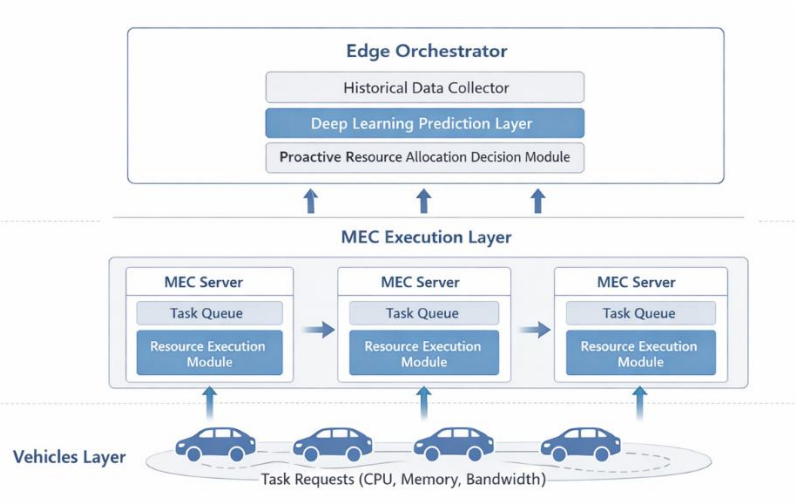

Figure 1 System architecture.

We consider a multi-edge computing (MEC) architecture composed of three logical layers: (i) Vehicular Layer, (ii) MEC Execution Layer, and (iii) Edge Orchestrator Layer. A set of vehicles $V = \{v_1, v_2, \ldots, v_N\}$ generates computation

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

tasks characterized by time-varying resource demands. These tasks are offloaded to a set of MEC servers $S = \{s_1, s_2, \ldots, s_M\}$, which execute them under limited computational capacity.

To enable proactive optimization, an intelligent Edge Orchestrator is deployed above the MEC servers. The orchestrator collects historical resource usage data from all MEC servers, predicts future workload using a Deep Learning (DL) model, and computes proactive resource allocation decisions, which are then transmitted back to the MEC layer.

An intelligent Edge Orchestrator is deployed above the MEC layer. It collects historical workload statistics to predict future resource demand using a DL model, then it computes proactive allocation decisions and sends allocation policies to MEC servers. This hierarchical separation ensures scalability and global workload awareness.

Workload Modeling

At time step t, the aggregated CPU demand arriving at server s_j is:

$$D_j^{cpu}(t) = \sum_{i \in V_j} d_i^{cpu}(t)$$

Similarly, the aggregated memory demand is:

$$D_j^{mem}(t) = \sum_{i \in V_j} d_i^{mem}(t)$$

where:

- $d_i^{cpu}(t)$ *and* $d_i^{mem}(t)$ present the CPU and memory requirements of vehicle v_i.
- Vj ⊆ V denotes vehicles associated with server s_j.

The workload vector is defined as:

$D_j(\mathrm{t}) = \{D_j^{cpu}(t), D_j^{mem}(t)\}$

Each server has finite capacity:

$C_j(\mathrm{t}) = \{C_j^{cpu}(t), C_j^{mem}(t)\}$

Although CPU saturation directly impacts processing delay, memory overload may trigger swapping or task rejection. Therefore, joint CPU–memory modeling provides a realistic representation of MEC constraints.

Deep Learning-Based Demand Prediction

To enable proactive allocation, the orchestrator learns the temporal dynamics of resource demand.

Given a historical observation window of length T :

$$Xj(t) = \{ Dj(t-T), \ldots, Dj(t-1) \}$$

The DL model approximates the mapping:

$$\widehat{D}_j(t) = f_\theta(X_j(t))$$

where:

- f_θ is the trained Deep Learning model,
- θ represents learnable parameters.

The model is trained by minimizing the Mean Squared Error (MSE):

$$L(\theta) = \frac{1}{K}\sum_{t=1}^{K}(D_j(t) - \widehat{D}_j(t))^2)$$

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

where K is the number of training samples.

Proactive Resource Allocation Strategy

Unlike conventional reactive systems, where allocation occurs after task arrival, the proposed framework performs proactive allocation based on predicted demand.

For each server s_j, the orchestrator computes:

$$A_j^{cpu}(t) = min\left(\hat{D}_j^{cpu}(t), C_j^{cpu}\right)$$

$$A_j^{mem}(t) = min\left(\hat{D}_j^{mem}(t), C_j^{mem}\right)$$

Overload is computed as:

$$O_j^{cpu}(t) = max\left(0, \hat{D}_j^{cpu}(t) - A_j^{cpu}(t)\right)$$

$$O_j^{mem}(t) = max\left(0, \hat{D}_j^{mem}(t) - A_j^{mem}(t)\right)$$

where:

- $A_j^{cpu}(t)\ and\ A_j^{mem}(t)$ is the allocated CPU and Memory resource
- $C_j^{cpu}\ and\ C_j^{mem}$ is the server capacity constraint.

Latency Modeling

In MEC systems, task latency is a critical performance metric, especially for real-time applications such as collision avoidance. The total latency of a task consists of the computation delay at the MEC server. In this work, latency is modeled as a function of the allocated resources and workload demand. Specifically, the latency of tasks processed at server s_j is approximated as:

$$Lj(t) = D_j(t) / A_j(t)$$

where $D_j(t)$ represents the CPU/Memory workload demand and $A_j(t)$ is the allocated CPU/Memory resource. This formulation reflects that higher allocated resources lead to lower processing delay, whereas insufficient allocation results in increased latency and potential SLA violations. This latency model is used to evaluate system performance and compute SLA violation rates.

Algorithm 1 describes the overall workflow of the proposed proactive resource allocation framework. At each time step, the Edge Orchestrator collects historical workload data from all MEC servers and constructs a temporal input sequence for prediction. The trained deep learning model (GRU) is then used to forecast the short-term CPU and memory demand for each server. Based on these predictions, the orchestrator computes proactive resource allocation decisions under capacity constraints. These allocation decisions are transmitted to the MEC servers before task arrival, enabling anticipatory provisioning and reducing the risk of congestion. Once tasks are executed, the system monitors overload conditions and measures task latency, which are used to evaluate system performance and update statistical metrics. This iterative process ensures continuous adaptation to dynamic workload variations and improves overall system responsiveness.

Algorithm 1: Proactive MEC Optimization via Edge Orchestrator

Input:

Historical workload data from MEC servers

Server capacities Cj

Trained DL model fθ

For each time step t do:

1. Collect historical resource usage Xj(t) from all MEC servers

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2. Predict future demand: $\widehat{D}_j(t) = f_\theta(X_j(t))$
3. Compute proactive allocation: $A_j(t) = min(\widehat{D}_j(t), C_j)$
4. Send allocation decisions to MEC servers
5. Execute tasks upon arrival
6. Compute overload Oj(t) and latency Lj(t)based on allocated resources
7. Update performance statistics

End For

Figure 2 illustrates the proposed hierarchical MEC optimization framework. Vehicles generate dynamic computational workloads that are offloaded to distributed MEC servers. Each MEC server executes tasks and periodically reports historical resource usage to a centralized Edge Orchestrator. The orchestrator employs a Deep Learning prediction module to estimate future workload and compute proactive resource allocation decisions. These decisions are transmitted back to the MEC servers before task arrival, enabling congestion mitigation and latency reduction. This separation between execution and intelligence ensures global workload awareness and scalable optimization.

Each vehicle is associated with the geographically closest MEC server based on Euclidean distance. Task offloading decisions follow a nearest-server policy, and handover occurs when vehicles move across coverage regions. This association mechanism allows modeling realistic load dynamics across distributed MEC servers.

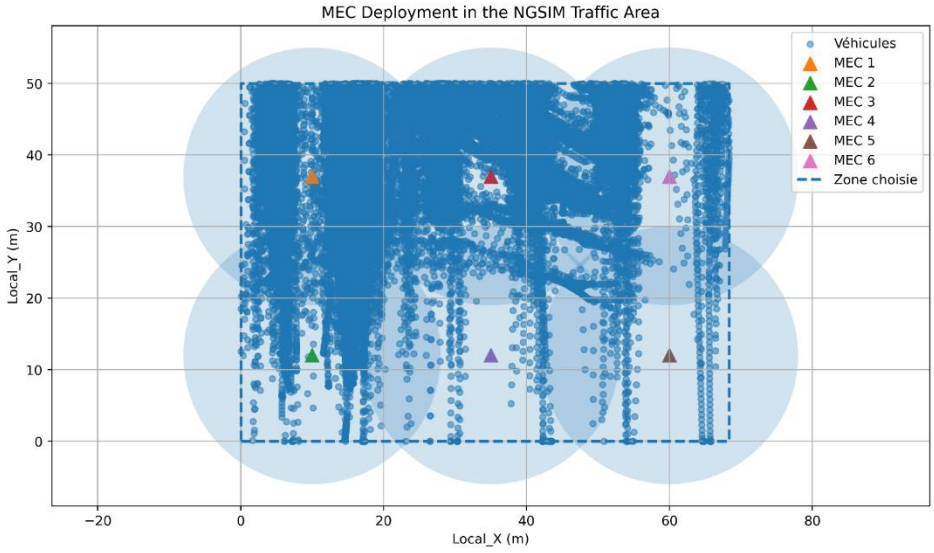

Figure 2 MEC Deployment in the NGSIM Traffic Area

RESULTS AND DISCUSSION

SLA Violation Metric

In latency-sensitive vehicular applications, such as collision avoidance, meeting strict delay constraints is critical for ensuring system reliability and safety. Therefore, we evaluate system performance using the Service-Level Agreement (SLA) violation rate. A task is considered to violate the SLA if its end-to-end execution latency exceeds a predefined threshold L_{max}. In this study, L_{max} is set to 100 ms, which is consistent with real-time vehicular safety requirements. Formally, the SLA violation rate is defined as:

$$SLA_{ViolationRate} = \left(\frac{\sum I\,(L_i > L_{max})}{N}\right)$$

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

where N is the total number of tasks, L_i represents the latency of task i, and I is an indicator function that equals 1 when the condition is true and 0 otherwise. This metric reflects the system's ability to satisfy real-time constraints and is used to evaluate the effectiveness of the proposed proactive resource allocation strategy.

System-Level Performance Evaluation

To evaluate the practical impact of the proposed approach, we compare its performance against two baseline strategies: static allocation (fixed resource provisioning without adaptation) and reactive allocation (resources allocated after task arrival). The evaluation focuses on three key system-level metrics: average latency, SLA violation rate, and resource CPU utilization.

The experiments were conducted using the NGSIM vehicular mobility dataset, which provides detailed trajectory information of vehicles on real highway segments. The dataset includes vehicle position, speed, and acceleration records sampled at 10 Hz. To simulate MEC workload dynamics, computational demand was modeled as a function of vehicle density and object detection tasks.

Simulation parameters include:

- Number of vehicles: 30
- Number of MEC servers: 6
- CPU capacity per server: 100 units
- Memory capacity: 64 GB
- GRU sequence length: 10
- Training epochs: 30

Convergence Analysis

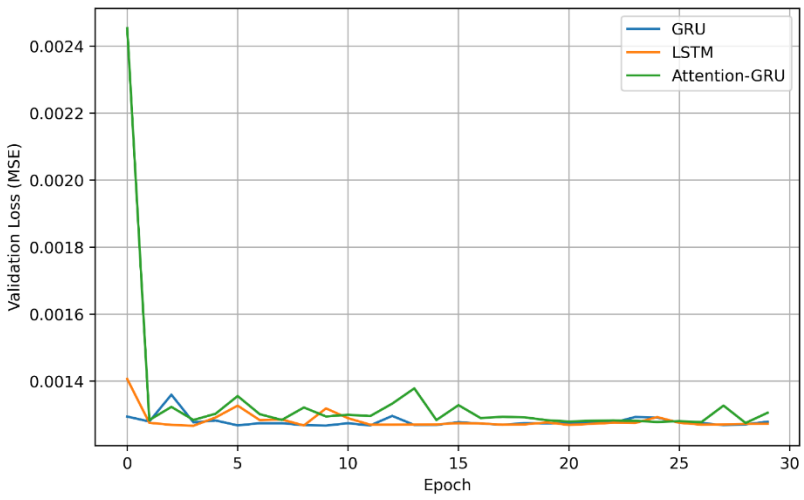

Figure 3 Validation Loss Comparison

Figure 3 presents the validation loss evolution of GRU, LSTM, and Attention-GRU models over 30 training epochs. All models demonstrate stable convergence behavior, with rapid loss reduction during the first few epochs followed by smooth stabilization. The three architectures reach convergence after approximately 10–15 epochs, indicating that the selected learning rate and sequence length are appropriate for the workload prediction task. No significant overfitting behavior is observed, as validation loss remains stable without divergence. Although Attention-GRU exhibits slightly higher initial fluctuations, its loss stabilizes similarly to that of standard recurrent architectures. Overall, the convergence curves confirm that all models learn the temporal workload dynamics effectively and stably.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

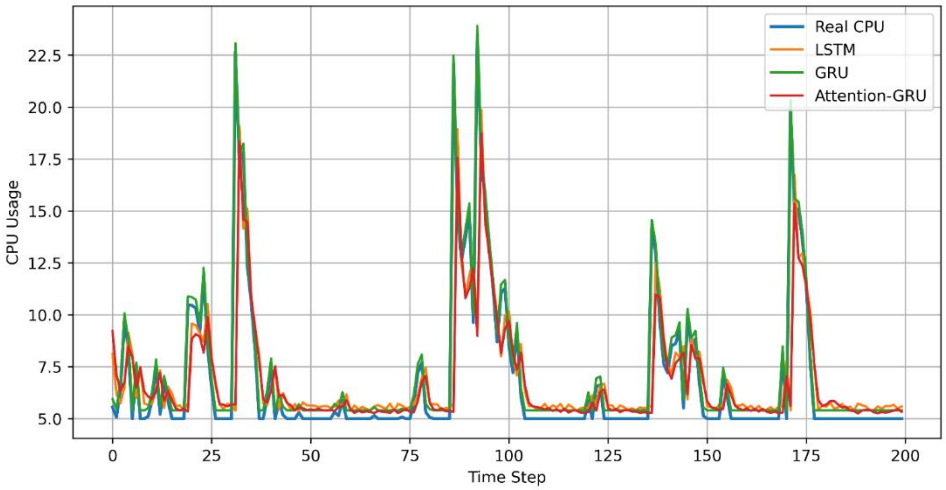

Figure 4 Comparison between predicted CPU usage for the different models

CPU Workload Prediction Performance

Figure 4 illustrates the real versus predicted CPU workload for a representative test segment. All three models successfully capture the general workload trend, including baseline levels and burst events. The prediction curves closely follow the ground truth signal, particularly during stable workload periods. Minor deviations are observed during sharp spike events, where models slightly underestimate peak magnitudes. This behavior is expected due to the stochastic burst generation and the inherent difficulty of modeling abrupt workload changes.

Among the three approaches, GRU achieves slightly better alignment during burst periods, LSTM demonstrates competitive tracking performance with smoother predictions, and Attention-GRU does not provide a visible improvement over the standard GRU. These visual observations are consistent with the quantitative evaluation.

Quantitative Evaluation

Table 1 summarizes the performance metrics on the test set. The results indicate that GRU achieves the lowest MSE and RMSE, LSTM achieves the lowest MAE, and Attention-GRU performs slightly worse than both baseline recurrent models.

Table 1: Performance comparison metrics on the test set.

Model	MSE	RMSE	MAE
GRU	3.1761	1.7100	0.5107
LSTM	3.2762	1.8073	0.5005
Attention-GRU	3.3056	1.8181	0.5365

However, the differences between GRU and LSTM are marginal (less than 0.3% relative difference in RMSE), suggesting that both architectures provide comparable prediction capability for multi-resource workload forecasting. The limited improvement of the Attention mechanism indicates that the temporal dependencies in this dataset may not require complex weighting strategies, as standard recurrent units already capture sufficient historical context.

Table 2: System-Level Performance Comparison

Method	Latency (ms)	SLA Violations (%)	CPU Utilization (%)
Reactive Allocation	120	18.2	65.1
Static Allocation	140	25.6	55.3
GRU	85	8.2	82.3

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

LSTM	96	10.7	78.5
Attention-GRU	94	11.2	79.1

The results demonstrate that the proposed proactive allocation strategy significantly improves system performance compared to conventional approaches. Specifically, the GRU-based method achieves the lowest latency (85 ms), representing a reduction of approximately 30% compared to reactive allocation and more than 38% compared to static allocation. In terms of SLA violations, the proposed approach reduces the violation rate to 8.2%, compared to 18.2% for reactive allocation and 25.6% for static allocation. This confirms the effectiveness of proactive workload prediction in maintaining real-time performance guarantees. Additionally, CPU utilization is significantly higher under the GRU-based strategy, reaching 82.3%, indicating better resource efficiency and reduced underutilization. Overall, these results highlight the advantage of integrating deep learning-based workload prediction with proactive resource orchestration in MEC environments.

Discussion

The experimental results highlight three key observations:

1. **Temporal modeling is effective:**

Both GRU and LSTM successfully model workload dynamics derived from vehicle density and mobility patterns.

2. **Marginal architectural differences:**

The performance gap between GRU and LSTM is minimal, indicating that increasing architectural complexity does not significantly enhance prediction accuracy in this scenario.

3. **Attention is not always beneficial:**

The Attention-GRU model does not outperform standard GRU, suggesting that the workload patterns exhibit relatively short- to mid-term dependencies that are already well captured by recurrent memory mechanisms.

From a practical MEC deployment perspective, GRU may be preferred due to its simpler structure and comparable accuracy, potentially resulting in lower computational overhead during online inference.

CONCLUSION

This paper presented a proactive deep learning-based resource allocation framework for multi-server MEC environments in connected vehicular networks. By leveraging a GRU-based prediction model, the proposed approach anticipates short-term workload variations and enables efficient resource provisioning before task arrival. Experimental results demonstrate that the proposed method achieves accurate workload prediction and significantly improves system-level performance, including reduced latency, lower SLA violation rates, and higher resource utilization compared to conventional reactive strategies. These results confirm the effectiveness of proactive AI-driven orchestration for real-time MEC applications. Future work will focus on integrating reinforcement learning for adaptive decision-making and extending the framework to large-scale vehicular scenarios.

REFRENCES

[1] A. Ismail, N. E. Khalifa, and R. A. El-Khoribi, “A survey on resource scheduling approaches in multi-access edge computing environment: A deep reinforcement learning study,” Cluster Computing, vol. 28, p. 184, 2025.
[2] R. Ke, “Resource allocation in Internet of Vehicles mobile edge computing scenarios,”
[3] Applied and Computational Engineering, vol. 167, pp. 94–99, 2025.
[4] T. Zheng, J. Wan, J. Zhang, et al, “Deep reinforcement learning-based workload scheduling for edge computing,” Journal of Cloud Computing, vol. 11, no. 3, 2022.
[5] L. Liu and Z. Xu, “Optimizing lightweight neural networks for efficient mobile edge computing,” Scientific Reports, vol. 15, p. 22056, 2025.
[6] M. Al Moteri, S. B. Khan, and M. Alojail, “Machine learning-driven ubiquitous mobile edge computing as a solution to network challenges in next-generation IoT,”
[7] Systems, vol. 11, no. 6, p. 308, 2023.

[8] M. Xie et al.,"Deep reinforcement learning-based computation offloading and distributed edge service caching for mobile edge computing," Computer Networks, vol. 250, p. 110564, 2024.
[9] H. Djigal, J. Xu, L. Liu, and Y. Zhang, "Machine and deep learning for resource allocation in multi-access edge computing: A survey," IEEE Communications Surveys & Tutorials, vol. 24, no. 4, pp. 2327–2360, 2022.
[10] Y. Zheng, T. Zhang, and J. Loo, "Dynamic multi-time scale user admission and resource allocation for semantic extraction in MEC systems," arXiv preprint arXiv:2307.02748, 2023.
[11] X. Ye, Y. Sun, and D. Wen, "End-to-end delay minimization based on joint optimization of DNN partitioning and resource allocation," arXiv preprint arXiv:2310.12937, 2023.
[12] S. Durga, E. Daniel, S. Deepakanmani, and V. K. Reshma, "Deep learning-based workload prediction and resource provisioning for mobile edge-cloud computing in healthcare applications," Sustainable Computing: Informatics and Systems, p. 101176, 2025.
[13] Z. Yu, X. Xu, and W. Zhou, "Task offloading and resource allocation strategy based on deep learning for mobile edge computing," Computational Intelligence and Neuroscience, 2022.
[14] Ullah, H. K. Lim, Y. J. Seok, and Y. H. Han, "Optimizing task offloading and resource allocation in edge-cloud networks: A deep reinforcement learning approach," Journal of Cloud Computing, 2023.
[15] Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in NOMA heterogeneous networks," IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 16001–16016, 2020.
[16] M. H. Adnan, Z. A. Zukarnain, and O. A. Amodu, "Fundamental design aspects of UAV-enabled MEC systems: A review on models, challenges, and future opportunities,"
[17] Computer Science Review, vol. 51, p. 100615, 2024.
[18] F. Jiang, K. Wang, L. Dong, C. Pan, and K. Yang, "Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale MEC networks,"
[19] IEEE Internet of Things Journal, vol. 7, no. 10, pp. 9278–9290, 2020.
[20] X. Wang, N. Cheng, L. Fu, W. Quan, R. Sun, Y. Hui, T. Luan, and X. S. Shen,
[21] "Scalable resource management for dynamic MEC: An unsupervised link-output graph neural network approach," in Proc. IEEE Int. Symp. Personal, Indoor and Mobile Radio Communications (PIMRC), 2023, pp. 1–6.
[22] T. Kamble, S. Deokar, V. Wadne, D. Gadekar, H. Vanjari, and P. Mange,
[23] "Predictive resource allocation strategies for cloud computing environments using machine learning," Journal of Electrical Systems, vol. 19, pp. 68–77, Dec. 2023.
[24] Brik and A. Ksentini, "Toward optimal MEC resource dimensioning for a vehicle collision avoidance system: A deep learning approach," IEEE Network, vol. 35, no. 3, pp. 74–80, 2021.
[25] U. K. Lilhore et al., "Optimizing energy efficiency in MEC networks: A deep learning approach with cybertwin-driven resource allocation," Journal of Cloud Computing, vol. 13, no. 1, p. 126, 2024.