Computer Networks 252 (2024) 110676

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Spatio-temporal graph learning: Traffic flow prediction of mobile edge computing in 5G/6G vehicular networks

Chao Song [a,*], Jie Wu [b], Kunyang Xian [a], Jianfeng Huang [a], Li Lu [a]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
[b] Department of Computer and Information Sciences, Temple University, United States of America

ARTICLE INFO

Dataset link: https://ieee-dataport.org/open-access/crawdad-romataxi

Keywords:
5G/6G vehicular networks
Graph neural network
Mobile edge computing

ABSTRACT

Mobile Edge Computing (MEC) is a key technology that emerged to address the increasing computational demands and communication requirements of vehicular networks. It is a form of edge computing that brings cloud computing capabilities closer to end-users, specifically within the context of vehicular networks, which are part of the broader Internet of Vehicles (IoV) ecosystem. However, the dynamic nature of traffic flows in MEC in 5G/6G vehicular networks poses challenges for accurate prediction and resource allocation when aiming to provide edge service for mobile vehicles. In this paper, we present a novel approach to predict the traffic flow of MEC in 5G/6G vehicular networks using graph-based learning. In our framework, MEC servers in vehicular networks are construed as nodes to construct a dynamic similarity graph and a dynamic transition graph over a duration of multiple days. We utilize Graph Attention Networks (GAT) to learn and fuse the node embeddings of these dynamic graphs. A transformer model is subsequently employed to predict the vehicle frequency accessing the edge computing services for the next day. Our experimental results have shown that the model achieves high accuracy in predicting edge service access volumes with low error metrics.

1. Introduction

With the rapid development of communication technology, 5G/6G networks have become increasingly prominent in various fields, especially in vehicular networks [1–3]. The application of 5G/6G in vehicular networks not only provides high-speed data transmission but also ensures the security and virtualization of information, which is crucial in the current intelligent and autonomous driving era. As a significant part of this advanced network, edge computing service plays an indispensable role [4,5]. Mobile Edge Computing (MEC) is a key technology that emerged to address the increasing computational demands and communication requirements of vehicular networks. It is a form of edge computing that brings cloud computing capabilities closer to end-users, specifically within the context of vehicular networks, which are part of the broader Internet of Vehicles (IoV) ecosystem. It allows for the processing of massive amounts of data generated by vehicles at the edge of the network, reducing latency, saving bandwidth, and enhancing user experience.

Traffic flow prediction is a critical component of MEC in vehicular networks. It allows for proactive resource management, improved service quality, and better support for the growing ecosystem of connected vehicle services. The integration of traffic flow prediction into MEC systems is a key enabler for the development of smart transportation systems and the realization of the full potential of the Internet of Vehicles (IoV). Fig. 1 illustrates an example of edge service nodes with visiting vehicles across a city map in 5G/6G vehicular networks. The time axis represents a series of days, and for each day, a snapshot is provided. However, accurately predicting the traffic flow for edge computing remains a challenge due to the dynamic nature of vehicular networks. The problem of traffic flow prediction for edge computing service refers to the challenge of accurately forecasting the resource requirements and service demands of edge computing applications in the context of vehicular networks. Varying mobility patterns and traffic scenarios make it difficult to accurately predict the computing demands of applications.

In vehicular networks, MEC servers exhibit spatio-temporal similarities based on the accessing vehicles. MEC servers that are geographically near to each other may serve similar types of vehicles and therefore have similar data patterns. For example, MEC servers on highways may receive more commercial vehicles and deliver similar kinds of high-speed-related services. Depending on the time of day or event, MEC servers may experience similar patterns in usage. For example, during rush hours, MEC servers in urban areas might process more

* Corresponding author.
E-mail addresses: chaosong@uestc.edu.cn (C. Song), jiewu@temple.edu.cn (J. Wu), xiankunyang@std.uestc.edu.cn (K. Xian), huangjianfeng@std.uestc.edu.cn (J. Huang), luli2009@uestc.edu.cn (L. Lu).

https://doi.org/10.1016/j.comnet.2024.110676
Received 15 March 2024; Received in revised form 22 May 2024; Accepted 26 July 2024
Available online 31 July 2024

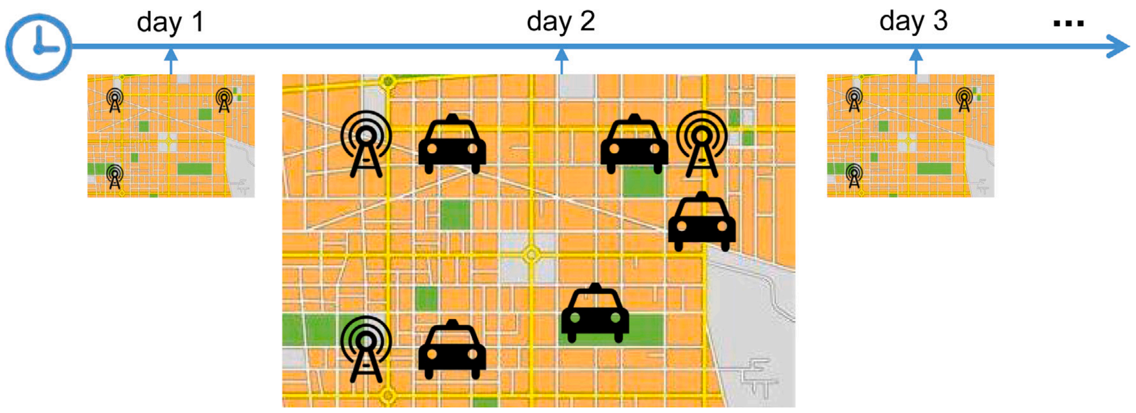


Fig. 1. An example of mobile edge computing in vehicular networks.

data related to route optimization to provide alternate routes and traffic updates. The traditional solution to predict the vehicular traffic flows uses time series prediction method. [6] models the univariate traffic condition data streams as seasonal ARIMA (AutoRegressive Integrated Moving Average) processes based on historical data. Regression models like linear regression, polynomial regression, and logistic regression can be used to establish a relationship between the dependent variable (demand) and one (or more) independent variables (features), in order to predict future demands. [7] proposes a spatio-temporal variable selection-based support vector regression (VS-SVR) model trained with the high-dimensional traffic data collected from all available road segments. Artificial Neural Networks can learn the relationship between inputs and outputs through training and are ideally suited for prediction tasks. Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are widely used to forecast edge computing service demands based on previous demands and other parameters. [8] proposes a CNN-based multi-feature predictive model that collectively predicts network-scale traffic flow with multiple spatio-temporal features.

In this paper, we propose an approach using graph learning to predict traffic flow of MEC in vehicular networks by considering both the spatial and temporal similarities. In the context of 5G/6G vehicular networks, to demonstrate the similarities among the edge service nodes, we construct a dynamic similarity graph over multiple days. To illustrate the temporal correlation between MEC servers, a dynamic transition graph is constructed, which captures the sequential interactions between nodes, representing the flow of vehicles from one edge service node to another over time. Then, we utilize Graph Attention Networks (GAT) to calculate two different node embeddings for these two dynamic graphs. These two node embeddings are fused together, and transmitted to a Transformer model for the traffic flow prediction of MEC in vehicular networks. Our experimental results demonstrate that the model achieves high accuracy in predicting edge service access volumes with low error metrics. Our contributions are multi-folds as follows:

- We analyze the spatio-temporal correlations among the MEC servers, and construct a dynamic similarity graph and a dynamic transition graph.
- We propose a GAT-Transformer method to process these dynamic graphs for the traffic flow prediction at MEC.
- We experiment the proposed method under the scenario with real-world data and compared with different algorithms.

The rest of the paper is organized as follows: Section 2 surveys the related work. Section 3 introduces preliminaries and the models of graphs. Section 4 presents the algorithms using GAT and Transformer. Section 5 evaluates our algorithms, and Section 6 concludes this paper.

2. Related work

5G/6G vehicular networks: A survey in [9] discusses current advancements in autonomous vehicles (AVs), automation levels, enabling technologies, and the requirements of 5G networks. It focuses on emerging technologies enabling the integration of 5G with AVs, the impact of 5G and beyond for AVs, and the envisaged security concerns in AVs. Sahrish Khan Tayyaba et al. in [10] address the limitations of current cellular technology and vehicular networks in satisfying the demands of vehicular network resource management. They discusses the application of machine learning for improving radio access in 5G vehicular networks. Xiang Cheng et al. in [11] introduce recent advanced techniques and important applications in vehicular communications and networking, focusing on the combination and integration of Vehicular Communication Networks (VCN) and connected vehicles. Md. Noor-A-Rahim et al. in [12] provide an overview of recent advances of machine learning in 6G vehicular networks, discussing the strengths, open challenges, maturity, and enhancing areas of these technologies. Hongzhi Guo in [2] predicts that 6G will be a key driving force for information interaction and social life after 2030, with characteristics such as AI-driven, highly dynamic, and extremely heterogeneous networks. Weijing Qi in [13] explores the concept of edge intelligence (EI) in 6G, combining artificial intelligence (AI) with mobile edge computing (MEC) to enhance the potential of edge-side data in vehicular networks.

Edge computing in vehicular networks: In the domain of vehicular networks, the integration of edge computing services has emerged as a pivotal approach to address the increasing computational demands of connected vehicles. The references provided offer a comprehensive overview of the current research and development in this field. The survey in [4] provides a detailed analysis of vehicular edge computing, emphasizing the benefits of mobile edge computing (MEC) in enhancing vehicular network performance. Hong Zhong et al. in [14] present secure edge computing framework for video reporting services in 5G-enabled vehicular networks, addressing critical security and privacy concerns. Another study [15] explores the application of distributed federated learning and network slicing for efficient resource allocation, aiming to improve quality of service (QoS). Lastly, [5] envisions the future of edge computing in 6G networks, focusing on the customization of services to meet the diverse and personalized needs of vehicles.

Machine learning (ML) in vehicular networks: The integration of machine learning (ML) in vehicular networks has emerged as a significant research area, offering innovative solutions to address the complex challenges of traffic management, security, and data processing works in this area range. From the comprehensive survey in [16] by Ye et al. which explores the potential of ML in solving various problems in vehicular networks. To the security-focused survey in [17] by Talpur and Gurusamy, which delves into the use of ML for enhancing security,

Table 1
Main notations.

Notation	Description
$v_i \in V$	The ith vehicle in the set.
$n = \|V\|$	Number of mobile vehicles.
$s_i \in S$	The jth MEC server in the set.
$m = \|S\|$	Number of MEC servers.
R	Wireless communication range of vehicle.
k	Number of days for historical data.
V_i^t	The visiting set of vehicles at MEC server s_i during the day t.
$G_S = \{G_S^1, G_S^2, \ldots\}$	The dynamic similarity graph.
$G_T = \{G_T^1, G_T^2, \ldots\}$	The dynamic transition graph.
τ_S, τ_T	The thresholds in similarity graph and transition graph.

these works highlight the versatility of ML in this domain. The research in [18] by Boukerche and Wang offers a detailed analysis of ML-based traffic prediction models, crucial for the development of Intelligent Transportation Systems. In [19], the research on IoV-based vehicular networks and the application of tree-based ML strategies for traffic management demonstrate the practical implications of ML in improving the efficiency and safety of vehicular networks.

Traffic flow prediction in vehicular networks: MEC provides computational resources at the edge of the network, close to the vehicles. Traffic flow prediction models can utilize these edge resources to process real-time data, such as vehicle positions, speeds, and routes, to predict future traffic conditions. This enables dynamic traffic management strategies, such as adjusting traffic signals or providing alternate route suggestions to drivers. Peng Sun et al. in [20] investigate the effectiveness of various machine learning based prediction models by considering both prediction accuracy and computational time cost, and present rigorous quantitative analysis to identify the important factors that may restrict the use of ML-based prediction models to support real-time services in the IoV environment. Hani M. Alnami et al. in [21] propose a system for predicting traffic flow in a VANET environment where vehicles on a highway segment form a cluster with a lead vehicle serving as the cluster head. The cluster head uses beacon information received from the vehicles in its cluster to determine the occupancy and the average speed in the corresponding segment and sends this information to the Roadside Unit (RSU) to be used in the prediction of abnormal traffic flow. Azzedine F. M. Boukerche et al. in [22] develop a traffic flow prediction solution consisting of three parts: a hybrid prediction model based on a Graph Convolutional Network (GCN) and a Recurrent Neural Network (RNN), which can extract spatial–temporal features from a dataset, a prediction strategy for multi-step prediction and an efficient training strategy for prediction on large-scale networks. Jian Chen et al. in [23] propose a graph convolution network based on node connection strength matrix to predict the traffic flow of the node, and a dynamics extractor for learning the various characteristics of traffic flow.

3. Spatio-temporal similarities of mobile edge computing in vehicular networks

In this section, we first introduce the scenario of mobile edge computing (MEC) in vehicular networks, and then discuss the spatio-temporal similarities of traffic flows among MEC servers. Last, we construct a dynamic similarity graph and a dynamic transition graph to model these spatio-temporal similarities. The main notations used in this paper are listed in Table 1.

3.1. Scenario

Mobile Edge Computing (MEC) is a key technology that has emerged to address the increasing computational demands and communication requirements of vehicular networks. It is a form of edge computing that brings cloud computing capabilities closer to end-users, specifically within the context of vehicular networks, which are part of the broader Internet of Vehicles (IoV) ecosystem. In vehicular networks, MEC enhances the performance of various applications by reducing latency, saving bandwidth, and improving the overall quality of service (QoS). MEC servers are deployed at the edge of the network, often in close proximity to vehicles, such as at road-side units (RSUs) or base stations. In the scenario of edge computing in vehicular networks, we have a network of n mobile vehicles equipped with wireless communication capabilities with a transmission range denoted by R which is set to 500 meters in our experiments [24,25]. These vehicles are constantly traveling and interacting with each other, forming a dynamic network topology. Additionally, we have m MEC servers strategically distributed throughout the network, which act as compute and storage resources.

Each vehicle in the network has a unique identifier, typically a Vehicle ID (VID). This ID is used to track the vehicle's interactions with the MEC servers. The MEC servers should be configured to capture and store access records. This includes setting up the necessary software and hardware components to handle the incoming data from vehicles. When a vehicle interacts with an MEC server, the server captures the Vehicle ID along with relevant metadata. This metadata may include the time of interaction, the type of service requested, the duration of the interaction, and any data transmitted. Each interaction is timestamped to record the exact time the vehicle accessed the MEC server. This is crucial for analyzing traffic patterns and ensuring that records are accurate. As shown in Fig. 2, the visiting sets of the two MEC servers are $\{v_1, v_2, v_3, v_4, v_5\}$ and $\{v_1, v_2\}$, respectively. The visiting set of MEC server s_i during the day t is the set of mobile vehicles which have accessed to s_i during the day t, and is denoted by V_i^t.

3.2. Motivation

For the purpose of investigating the interconnectivity among edge service nodes within vehicular networks leveraging edge computing, we have the CRAWDAD roma/taxi dataset[1] for our analysis. This dataset meticulously records the movement patterns of taxis within the city of Rome, providing a rich source of real-world data for our study. Each entry in the dataset corresponds to a taxi's location at a specific timestamp. The trajectories are sorted by timestamp, offering a temporal sequence of the taxis' positions. Fig. 3(a) shows the records of these taxis' locations. We randomly select m taxis' positions as the locations of MEC servers. This dataset serves as a valuable resource for our research, as it allows us to analyze the spatial and temporal dynamics of vehicular movement in a dense urban environment.

3.2.1. Spatial similarity

Spatial similarity between MEC servers refers to the geographical proximity of these servers in the network. Two MEC servers are considered spatially similar if they are located physically close to each other. It is assumed that neighboring servers may have similar characteristics due to shared environmental factors and similar user behaviors. Thus, spatial similarity can be valuable in predicting accessing frequency.

In our study, we conducted an analysis to explore the correlation between the distance of edge service nodes and the similarity of vehicles visiting them daily. Utilizing the dataset, we aggregated the visiting sets of MEC servers on a daily basis and calculated the Jaccard index or Jaccard similarity [26] among them. Additionally, we measured the pairwise distances between edge service nodes. By compiling these records over 30 days, we categorized the distances into several intervals and analyzed the distribution of similarity across these intervals, with distance as the x-axis and similarity as the y-axis. The results under 20 MEC servers and 50 MEC servers are visualized using a box-plot as shown in Figs. 3(b) and 3(d), respectively. Each box represents the distribution of similarity within a specific distance interval. The black

[1] https://ieee-dataport.org/open-access/crawdad-romataxi

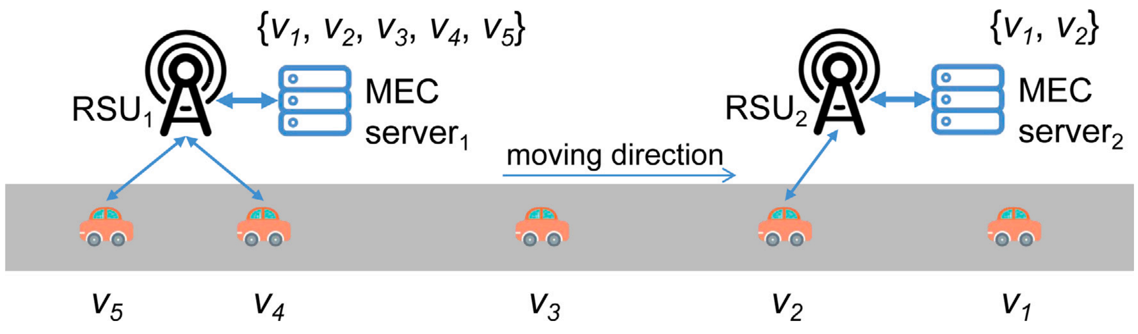


Fig. 2. The scenario of a vehicular network with edge computing.

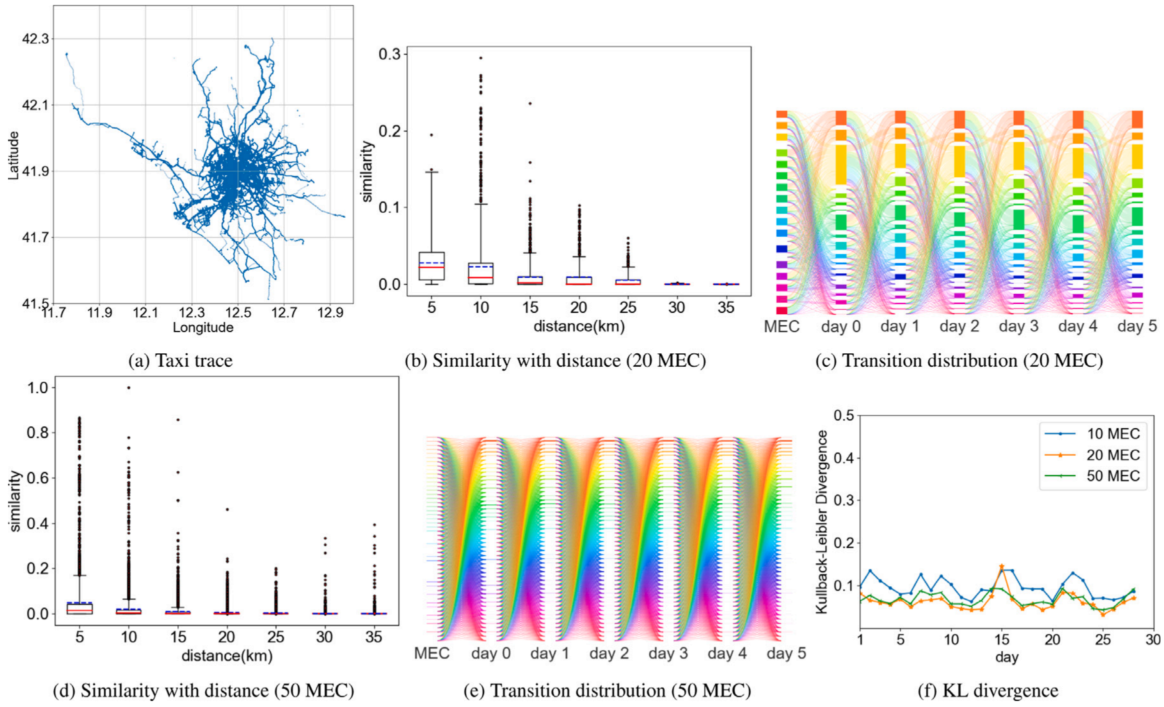


Fig. 3. Spatio-temporal similarities of traffic flows at MEC.

dots within the boxes indicate outliers, which are instances of unusually high similarity. The bottom and top of each box correspond to the first (Q1) and third (Q3) quartiles of the data, respectively, while the whiskers extend to the maximum and minimum values. The median is represented by the red line, and the mean is indicated by the blue line.

According to the results under 20 and 50 MEC servers, we notice that as the distance between MEC servers increases, the similarity of the vehicles visiting them tends to decrease. This trend suggests that vehicles are more likely to visit MEC servers that are geographically closer to each other, which is consistent with the expectation that proximity facilitates easier and more frequent interactions. This insight is valuable for network planners and service providers, as it can guide the strategic placement of MEC servers to optimize network efficiency and enhance the user experience in vehicular networks. By understanding the relationship between distance and similarity, we can better predict vehicle movement and edge services according to the needs of the vehicular ecosystem.

3.2.2. Temporal similarity

Temporal similarity among MEC servers refers to the similarity of their services pattern over time. In other words, two MEC servers are considered to have temporal similarity if their pattern of providing services to mobile vehicles follows a similar trend over time. In our research, we have delved into the temporal correlations between edge service nodes within vehicular networks by analyzing the dataset provided. We focused on the probability distribution of vehicle transitions from one edge service node to the next, aggregating the data on a daily basis to observe the distribution of traffic flow between these nodes. To visualize this data, we employed a Sankey diagram shown in Figs. 3(c) and 3(e), which illustrates the flow of traffic between nodes over a period of six consecutive days with 20 and 50 MEC servers, respectively. The diagram arranges the m MEC servers in a columnar format, with each server represented by a distinct color. The lines connecting the nodes across columns signify the traffic flow on a given day, thus providing a comprehensive view of the transition patterns over the six-day period. The Sankey diagram reveals a consistent pattern in the traffic flow between edge service nodes on a daily basis. This consistency suggests that there are predictable and recurring traffic patterns within the vehicular network, which could be attributed to factors such as regular commuting routes, event-driven traffic, or the inherent structure of the road network.

Moreover, we calculate the Kullback–Leibler (KL) divergence [27] to analyze the transition probabilities between edge service nodes over time. For each day, a transition probability matrix is constructed to represent the likelihood of vehicles moving from one edge service node to another. This is done by calculating the probability of transition for each pair of nodes, considering the traffic data collected for that day. The resulting matrix for day t is denoted as P_t. The KL divergence, a measure of the difference between two probability distributions, is computed between consecutive days' transition probability matrices. Specifically, the KL divergence between P_t and P_{t+1} is calculated as $D_{KL}(P_t \parallel P_{t+1})$. This value quantifies how much one distribution diverges from the other, which in this context, reflects the change in traffic patterns from one day to the next. A line graph is then created to visualize the KL divergence over time. The horizontal axis (x-axis) represents the days, and the vertical axis (y-axis) represents the Kullback–Leibler Divergence values. Each point on the graph corresponds to the KL divergence for a specific day, providing a visual representation of the changes in traffic transitions. The results of the KL divergence over time are shown in Fig. 3(f). The average KL divergences under 10, 20 and 50 MEC servers are 0.096, 0.062 and 0.068, respectively. We notice that the low values of KL divergences imply similar distributions of transition probabilities over time, which reveal the temporal similarities of traffic flows at the MEC servers over multiple days.

3.3. Dynamic similarity graph

To model the similarity between edge service nodes, we construct a dynamic similarity graph, denoted as $G_S = (S, E_S)$, where S and E_S represent the sets of vertexes and edges. The dynamic similarity graph is composed of daily similarity subgraphs, where the tth day's subgraph is represented by G_S^t. The collection of these subgraphs over multiple days forms the dynamic similarity graph, expressed as $G_S = \{G_S^1, G_S^2, \dots\}$. For each subgraph G_S^t, the vertexes represent the MEC servers, and the value associated with each vertex s_i is the frequency of visits by vehicles to that MEC server on day t, i.e., $|V_i^t|$. The Jaccard Similarity J_{ij}^t between any two vertexes s_i and s_j on day t is computed using the following formula: $J_{ij}^t = \frac{|V_i^t \cap V_j^t|}{|V_i^t \cup V_j^t|}$. Here, $|\cdot|$ denotes the cardinality of a set. An edge e_{ij}^t is established between nodes s_i and s_j in G_S^t if their Jaccard Similarity exceeds a predefined threshold τ_S: $e_{ij}^t \in E_S^t$ if $J_{ij}^t > \tau_S$. The weight w_{ij}^t of the edge e_{ij}^t is set to the Jaccard Similarity value: $w_{ij}^t = J_{ij}^t$.

By aggregating these daily subgraphs, we create the dynamic similarity graph G_S, which encapsulates the evolving relationships between edge service nodes based on vehicle visitation patterns. The algorithm that builds the dynamic similarity graph is a step-by-step process designed to construct a graph that represents the similarity between edge service nodes in a vehicular network over multiple days. As shown in Algorithm 1, the algorithm begins by initializing an empty dynamic similarity graph G_S, which will accumulate the similarity subgraphs for each day. The algorithm iterates over each day t from 1 to the total number of days. For each day, it initializes an empty similarity subgraph G_S^t. For the current day t, the algorithm retrieves the set of edge service nodes S that were active or had vehicle visits. The algorithm then calculates the Jaccard Similarity J_{ij}^t for each pair of nodes s_i and s_j. This similarity measure is based on the intersection and union of the sets of vehicles that visited each pair of nodes. If the calculated Jaccard Similarity exceeds a predefined threshold τ_S, the algorithm adds a directed edge between the two nodes in the subgraph G_S^t. The weight of this edge is set to the calculated similarity value. After processing all pairs of nodes for a given day, the algorithm adds the daily subgraph G_S^t to the overall dynamic similarity graph G_S. The algorithm increments the day counter t and repeats the process for each day until all days have been processed. Finally, the algorithm returns the complete dynamic similarity graph G_S, which now contains the similarity relationships between edge service nodes across all days. This constructed dynamic graph is useful for analyzing the patterns of vehicle visits to edge service nodes and can help in understanding the structure and dynamics of vehicular networks, which is essential for traffic prediction, resource allocation, and network optimization. Fig. 4 shows the constructed dynamic similarity graph of the dataset with 50 MEC servers in 5 days, where the threshold τ_S is set to 0.01. The distribution of edge weights for each subgraph is also shown in Fig. 4 with PDF (Probability Density Function), and the distribution observed also appears to be similar to a power-law distribution. Moreover, the curve decays more rapidly than dynamic similarity graph.

Algorithm 1: Algorithm for constructing dynamic similarity graph

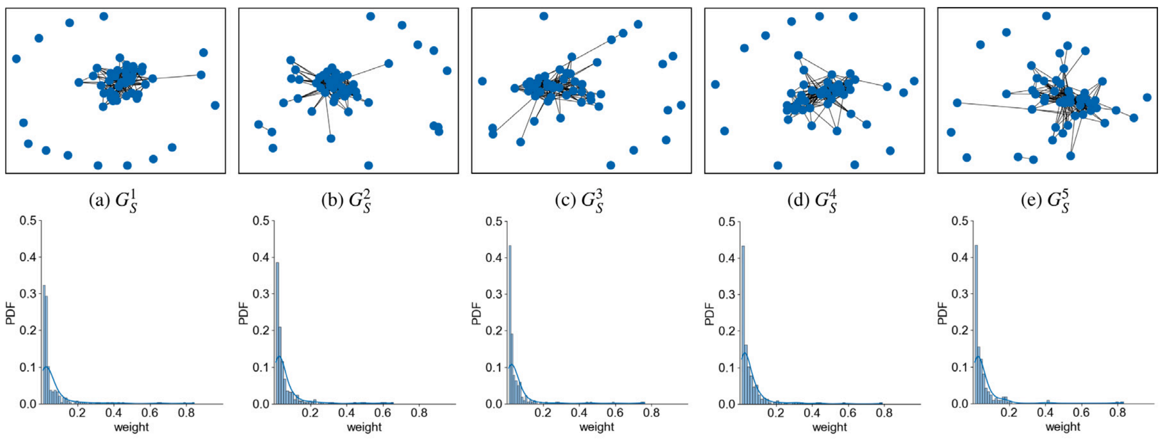
```
Input: Set of MEC servers $S$, total number of days $T$, threshold $\tau_S$
Output: Dynamic similarity graph $G_S$
while $t \le T$ do
    $G_S^t \leftarrow \emptyset$
    $V_t \leftarrow GetNodesForDay(S, t)$ for $i \in [1, |S|]$ do
        for $j \in [i+1, |S|]$ do
            $J_{ij} \leftarrow CalculateSimilarity(s_i, s_j)$
            if $J_{ij} > \tau_S$ then
                $G_S^t \leftarrow G_S^t \cup \{(s_i, s_j, J_{ij})\}$
            end
        end
    end
    $G_S \leftarrow G_S \cup G_S^t$
    $t \leftarrow t + 1$
end
```

3.4. Dynamic transition graph

To illustrate the temporal correlation between MEC servers, a dynamic transition graph $G_T = (S, E_T)$ is constructed, where S and E_T are the sets of vertexes and edges. The dynamic transition graph captures the sequential interactions between nodes, representing the flow of vehicles from one edge service node to another over time.

The dynamic transition graph is composed of daily transition subgraphs, where the tth day's subgraph is denoted as G_T^t. The collection of these subgraphs forms the dynamic transition graph, expressed as $G_T = \{G_T^1, G_T^2, \dots\}$. For each subgraph G_T^t, the vertexes represent the MEC servers. A directed edge e_{ij}^t from vertex s_i to vertex s_j indicates that vehicles have consecutively visited node v_i and then node v_j, and the number of visited vehicles should be larger than a threshold τ_T. The weight w_{ij}^t of this directed edge represents the number of vehicles that have made this transition.

The algorithm for building a dynamic transition graph is designed to capture the temporal relationships between edge service nodes in a vehicular network by tracking the sequence of visits made by vehicles. This algorithm is particularly useful for understanding how traffic flows from one service node to another over time, which can inform network planning and optimization. As shown in Algorithm 2, the algorithm starts by initializing an empty dynamic transition graph G_T, which will be composed of daily transition subgraphs. The algorithm loops through each day t from 1 to the total number of days. For each day, it initializes an empty subgraph G_T^t and a set to track vehicle transitions. For the current day t, the algorithm retrieves the set of edge service nodes S that are active or have been visited by vehicles. The algorithm tracks the transitions between edge service nodes, recording the pairs of nodes (s_i, s_j) where vehicles have moved from s_i to s_j. For each observed transition, the algorithm checks if a directed edge from s_i to s_j already exists in the subgraph G_T^t. If not, it adds the edge with an initial weight of 1. If the edge exists, it increments the weight by 1 to reflect the additional transition. It checks all the edges in the subgraph G_T^t and deletes any edges with weights lower than the threshold τ_T. After processing the transitions for a day, the algorithm adds the daily subgraph G_T^t to the overall dynamic transition graph G_T. The algorithm increments the day counter t and repeats the process for each day until all days have been processed. Finally, the algorithm returns the complete dynamic transition graph G_T, which now contains the directed edges and their weights, representing the



(a) G_S^1 (b) G_S^2 (c) G_S^3 (d) G_S^4 (e) G_S^5

(f) Weights of G_S^1 (g) Weights of G_S^2 (h) Weights of G_S^3 (i) Weights of G_S^4 (j) Weights of G_S^5

Fig. 4. An example of a dynamic similarity graph with 50 MEC servers over 5 days ($\tau_S = 0.01$).

flow of vehicles between edge service nodes across all days. Fig. 5 shows the constructed dynamic transition graph of the dataset with 50 MEC servers over 5 days, where the threshold τ_T is set to 5. The distribution of edge weights for each subgraph is also shown in the figure with PDF (Probability Density Function), and the distribution observed appears to be similar to a Power-Law Distribution.

Algorithm 2: Algorithm for constructing dynamic transition graph

```
Input: Set of MEC servers $S$, total number of days $T$, threshold $\tau_T$
Output: Dynamic transition graph $G_T$
while $t \le T$ do
    $G_T^t \leftarrow \emptyset$
    $S_t \leftarrow GetNodesForDay(S, t)$
    $transitions \leftarrow \emptyset$
    $TrackVehicleTransitions(S_t, transitions)$ for $(s_i, s_j) \in transitions$ do
        if $(s_i, s_j) \notin E_T^t$ then
            $E_T^t \leftarrow E_T^t \cup \{(s_i, s_j)\}$
        end
        $UpdateEdgeWeights((s_i, s_j), transitions)$
    end
    for $(s_i, s_j) \in E_T^t$ do
        if $Weight(s_i, s_j) < \tau_T$ then
            Delete $(s_i, s_j) \in E_T^t$
        end
    end
    $G_T \leftarrow G_T \cup G_T^t$
    $t \leftarrow t + 1$
end
```

4. Traffic flow prediction algorithm with dynamic graphs

In this section, we first present the proposed traffic flow prediction algorithm with dynamic graphs. Then, we introduce the details of graph learning with GAT and a Transformer-based prediction model. Finally, we discuss the interpretability of the proposed models.

4.1. Overview of the proposed traffic flow prediction algorithm

The traffic prediction algorithm for 5G/6G vehicular networks utilizes two dynamically constructed graphs to capture the complex interactions between edge service nodes: a dynamic similarity graph and a dynamic transition graph. These graphs are instrumental in understanding the network's structure and the flow of vehicles. As shown in Fig. 6, the algorithm begins by building the dynamic similarity graph, which represents how similar the traffic patterns are between different edge service nodes. This is achieved by calculating the Jaccard Similarity based on the vehicles that visit each node daily. A dynamic transition graph is also constructed to reflect the temporal transitions of vehicles from one edge service node to another, indicating the direction and frequency of these movements. Using Graph Attention Networks (GAT) [28], the algorithm computes node embedding for both graphs. These embedding are essentially feature representations of each node that capture its importance and relationships within the graph. The GAT assigns different attention weights to the neighboring nodes, allowing the model to focus on the most relevant connections.

Once the node embeddings are obtained, they are concatenated to form a comprehensive feature vector for each edge service node. The final step involves using a Transformer model [29] to predict the traffic for the next day. The Transformer, known for its ability to handle long sequences and capture dependencies, processes the feature vectors to forecast the number of vehicles that will visit each edge service node. Therefore, the algorithm combines graph-based features with the power of the Transformer to create a robust traffic prediction model for vehicular networks.

4.2. Graph learning with GAT

Graph Attention Networks (GATs) [28] are a class of Graph Neural Networks (GNNs) designed to operate on graph-structured data. They leverage attention mechanisms to weigh the importance of each node's neighbors, allowing the model to focus on the most relevant information.

Each node s_i in the graph is represented by a feature vector $\mathbf{h}_i \in \mathbb{R}^F$, where F is the dimensionality of the feature space. A shared linear transformation is applied to the input features. This is done using a weight matrix $\mathbf{W} \in \mathbb{R}^{F' \times F}$, where F' is the dimensionality of the output features. The transformed features are denoted as $\mathbf{Wh}_i$. The attention mechanism computes the importance of each neighbor's features for a given node. For node s_i and its neighbor s_j, the attention coefficient α_{ij} is calculated using a feedforward neural network with a single hidden layer: $\alpha_{ij} = \text{softmax}\left(\frac{\exp(\mathbf{a}^T[\mathbf{Wh}_i \oplus \mathbf{Wh}_j])}{\sum_{k \in \mathcal{N}(i)} \exp(\mathbf{a}^T[\mathbf{Wh}_i \oplus \mathbf{Wh}_k])}\right)$. Here, $\mathbf{a}$ is a learnable weight vector, $\oplus$ denotes concatenation, and $\mathcal{N}(i)$ is the neighborhood of node s_i. The softmax function ensures that the attention coefficients sum to 1.

Multi-head attention is an extension of the basic attention mechanism where the input is processed through multiple attention layers (heads) in parallel. Each head learns to focus on different aspects or representations of the input data. The node embedding for s_i is computed as a weighted sum of its neighbors' features, using the attention coefficients: $\mathbf{e}_i = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{Wh}_j\right)$. The function σ is an activation function, such as the LeakyReLU. GATs often employ multi-head attention, where multiple attention mechanisms are applied independently, and their outputs are concatenated or averaged.

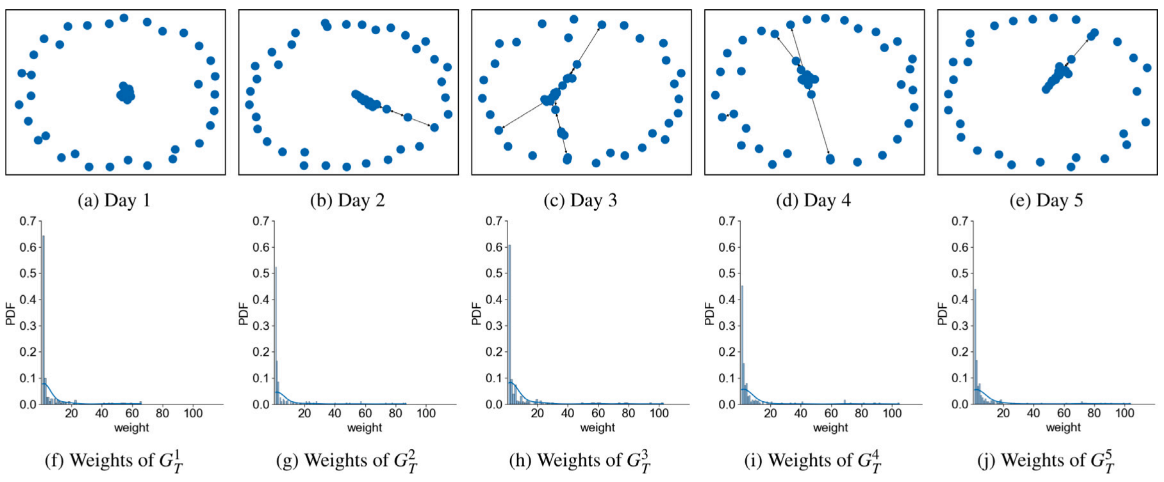


Fig. 5. An example of a dynamic transition graph with 50 MEC servers over 5 days ($\tau_T = 5$).

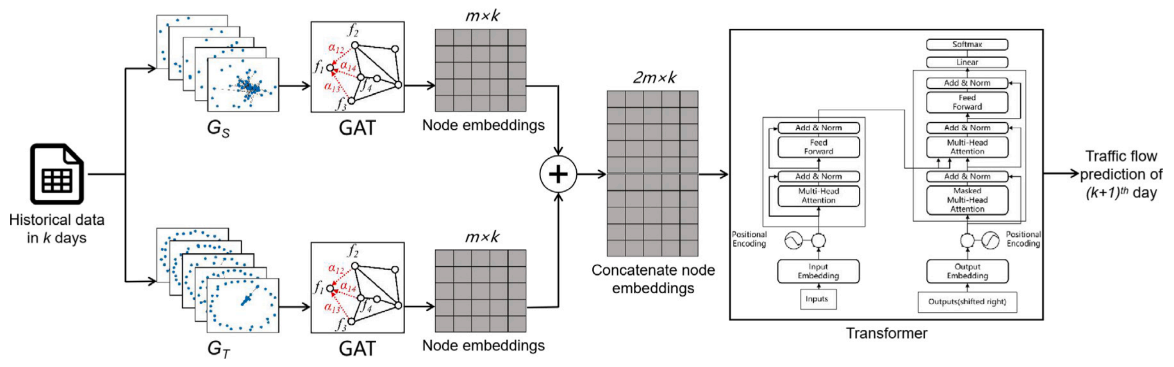


Fig. 6. Flowchart of traffic flow prediction algorithm.

This allows the model to learn multiple representations of the graph: $\mathbf{e}_i = \text{concat}\left(\{\sigma\left(\sum_{j\in\mathcal{N}(i)}\alpha_{ij}^{(k)}\mathbf{W}^{(k)}\mathbf{h}_j\right)\}_{k=1}^{K}\right)$. Here, K is the number of attention heads, and $\alpha_{ij}^{(k)}$ and $\mathbf{W}^{(k)}$ are the attention coefficients and weight matrices for the kth head. Multiple GAT layers can be stacked to learn deeper representations. The output of one layer becomes the input to the next: $\mathbf{h}_i' = \text{GATLayer}(\mathbf{h}_i, \mathbf{A})$. Here, $\mathbf{A}$ is the adjacency matrix of the graph. The attention heads allow the network to learn multiple attention functions over the nodes of a graph. This means that for each node, the GAT can learn to weigh the importance of its neighbors differently depending on the head.

By applying GAT to both the dynamic similarity graph and the dynamic transition graph, we can obtain two different node embeddings that capture the spatio-temporal relationships within the vehicular network. These embedding can then be used as input to a Transformer model for traffic prediction tasks.

4.3. Transformer-based prediction model

The Transformer model [29], originally designed for natural language processing tasks, has been adapted for various sequence modeling problems, including traffic prediction in vehicular networks. The input to the Transformer model is structured as a tensor with dimensions $[batch\_size, sequence\_length, feature\_size]$ representing the historical traffic data for each edge service node. As shown in Fig. 6, this tensor with dimensions $[1, k, 2m]$ is organized with a concatenate node embedding learning from the dynamic similarity graph and the dynamic transition graph, where k is number of days in historical data and m is the number of MEC servers. $\mathbf{x}_i$ represents the feature vector for node s_i at day t, which includes the node's embedding from the dynamic similarity graph and dynamic transition graph, as well as additional features such as historical visit frequency. Since the Transformer does not have a built-in notion of order, positional encodings are added to the input feature vectors to provide information about the position of each node in the sequence. The positional encoding is typically a fixed function of the position index, which can be learned or determined by a predefined function, such as a sine or cosine function.

The input feature vectors $\mathbf{x}_i$ are passed through an embedding layer to obtain the input embedding $\mathbf{e}_i$. This layer can be shared across all positions. The Transformer model applies self-attention to the input embedding. The self-attention mechanism computes the attention scores for each pair of positions in the sequence, allowing the model to weigh the importance of each input vector relative to the others: $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$. Here, Q, K, and V are the query, key, and value matrices derived from the input embedding. d_k is the dimensionality of the key vectors. The Query Q represents the current element that the model is focusing on. The Key K corresponds to the elements that the Query is compared against to calculate the attention scores. The Value V represents the elements that contribute to the construction of the final output after the attention scores have been computed and applied. After the self-attention step, layer normalization and residual connections are applied to stabilize the training process and allow the model to learn more effectively: $\mathbf{e}_i' = \text{LayerNorm}(\mathbf{e}_i + \text{Attention}(\mathbf{e}_i, \mathbf{e}_i, \mathbf{e}_i))$.

Each attention output is passed through a feed-forward network, which consists of two linear layers with a ReLU activation function in between: $\mathbf{e}_i'' = \text{ReLU}\left(\mathbf{W}_2\text{ReLU}\left(\mathbf{W}_1\mathbf{e}_i' + \mathbf{b}_1\right) + \mathbf{b}_2\right)$. Here, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{b}_1$,

Table 2
Parameters of models.

Model	Parameter	Value	Description
GAT	in_feats	1	Size of each input sample.
	h_feats	64	Size of each hidden sample.
	out_feats	1	Size of each output sample.
	layers	2	Number of message passing layers.
LSTM	input_size	20/50/100	The number of expected features in the input.
	output_size	1	The size of output features from the last layer of the LSTM.
	hidden_size	128	The number of features in the hidden state.
	num_layers	3	Number of recurrent layers.
Transformer	input_size	20/50/100	The size of each input sample.
	d_model	64	The number of expected features in the encoder/decoder inputs.
	output_size	20/50/100	The size of each output sample.
	seq_len	7/14	The length of time sequence.
	nhead(encoder)	8	The number of heads in the multi-head-attention models.
	layers(encoder)	5	The number of sub-encoder-layers in the encoder.

and $\mathbf{b}_2$ are learnable parameters. The final output of the Transformer model is a sequence of embeddings that represents the input data in a compressed form. For traffic prediction, a linear layer is applied to the output embedding to generate the predicted traffic flow for each node: $\hat{y}_i = \mathbf{W}_y \mathbf{e}_i'' + b_y$. Here, $\mathbf{W}_y$ and b_y are the weight matrix and bias for the output layer, and $\hat{y}_i$ is the predicted traffic flow for node v_i. The model's predictions are compared to the actual traffic data using a loss function, such as mean squared error (MSE), to measure the performance of the model during training: $\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$. Here, m is the number of nodes (MEC servers), y_i is the actual traffic flow, and $\hat{y}_i$ is the predicted traffic flow.

5. Experimental evaluations

In this section, we first introduce the experimental setup, compared methods and metrics. Then, we discuss the results for the impact of the parameter learning rate, and ablation study of different methods.

5.1. Experimental setup

5.1.1. Dataset

The experimental setup for traffic flow prediction at MEC in 5G/6G vehicular networks utilizes the CRAWDAD roma/taxi dataset, which provides a rich source of real-world taxi trip data. Here's an overview of the dataset configuration for the experiment: The CRAWDAD roma/taxi dataset is chosen for its comprehensiveness and relevance to vehicular network traffic patterns. This dataset typically includes information such as taxi pickup and drop-off locations, timestamps, and trip durations. The dataset spans a total of 30 days, offering a sufficient time frame to capture the dynamics of vehicular traffic and to train and test the prediction model. Within the dataset's time range, 20 or 50 or 100 MEC servers are randomly generated. These nodes represent potential points of interest or service areas within the vehicular network. To ensure that the model is not biased by the scale of the input features, Min–Max normalization is applied to both the training and testing datasets. This process scales the data to a range between 0 and 1, which can improve the convergence of the model during training.

5.2. Compared methods

In the experimental evaluation of our traffic prediction algorithm for 5G/6G vehicular networks, we compare the proposed Transformer-related models with the Long Short-Term Memory (LSTM) related model. We provide an overview of the comparative methods used in the study:

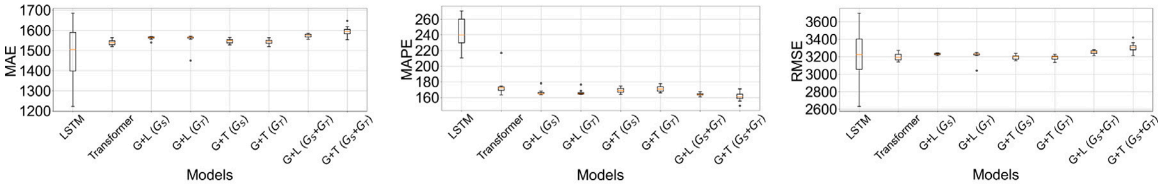
- **LSTM**: This method utilizes the LSTM model alone to predict the traffic flow at edge service nodes [30,31].
- **Transformer**: Similarly, this approach employs the Transformer model exclusively for traffic flow prediction [32,33].
- **GAT+LSTM** (G_S) or **G+L** (G_S): This method combines the GAT model with the LSTM model, using the dynamic similarity graph as input to predict traffic flow. The GAT provides node embeddings that capture the similarity between nodes, which are then processed by the LSTM for temporal prediction.
- **GAT+LSTM** (G_T) or **G+L** (G_T): Similar to the previous method, but this time the dynamic transition graph is used as input. This graph represents the temporal transitions between nodes, providing a different perspective on the network's structure.
- **GAT+Transformer** (G_S) or **G+T** (G_S): The GAT model is paired with the Transformer model, feeding the dynamic similarity graph's node embeddings into the Transformer for sequence-based prediction.
- **GAT+Transformer** (G_T) or **G+L** (G_S+G_T): This approach uses the dynamic transition graph's node embeddings as input to the Transformer model, focusing on the temporal transitions for prediction.
- **GAT+LSTM** (G_S+G_T) or **G+L** (G_S+G_T): This method leverages both the dynamic similarity graph and the dynamic transition graph by combining them into input for the LSTM model. This dual-input approach aims to capture both the similarity and transition aspects of the network.
- **GAT+Transformer** (G_S+G_T) or **G+T** (G_S+G_T): The final method in our comparison combines the strengths of both GAT-generated graphs by using them together as input to the Transformer model. This comprehensive approach seeks to integrate multiple facets of the network's structure for improved predictive accuracy.

5.2.1. Parameters of models

All models adopt the random seed setting of torch.manual_ seed(12345). The division ratio for all training sets, validation sets, and test sets is 6:2:2. The parameters of models are listed in Table 2.

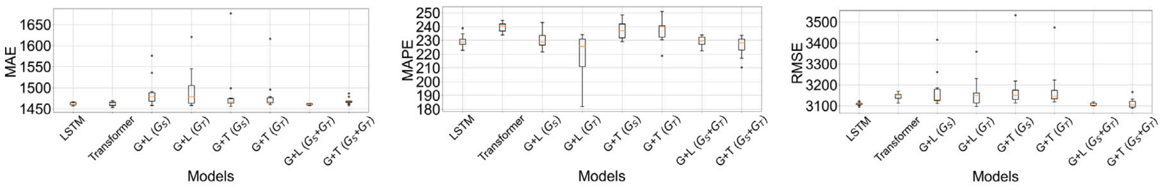
The GAT model consists of two layers. The input feature dimension (in_feats) is set to 1, and the output feature dimension (out_feats) is set to 1. The dimensions of the features in the hidden layers (h_feats) is set to 64. The GAT model employs a multi-head attention mechanism with four heads.

The number of layers in the LSTM prediction model is 3. The number of expected features in the input data for each time step ($input\_size$) is set to 20/50/100. The number of output features produced by the LSTM at each time step ($output\_size$) is set to 1. The number of features in the hidden state of the LSTM ($hidden\_size$) is set to 128. The model is trained to minimize the Mean Squared Error (MSE) loss function.



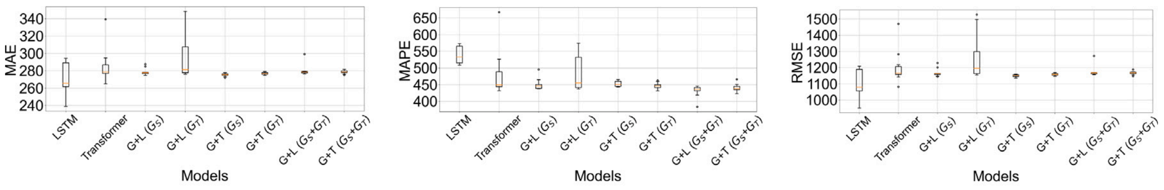
(a) MAE (b) MAPE (c) RMSE

Fig. 7. Errors of the compared models under 20 MEC servers for 7+1 prediction, $lr \in [0.001, 0.01]$.



(a) MAE (b) MAPE (c) RMSE

Fig. 8. Errors of the compared models under 20 MEC servers for 14+1 prediction, $lr \in [0.001, 0.01]$.



(a) MAE (b) MAPE (c) RMSE

Fig. 9. Errors of the compared models under 100 MEC servers for 7+1 prediction, $lr \in [0.001, 0.01]$.

The transformer prediction model parameters are as follows: The dimensionality of the input embeddings ($input\_size$) is set to 20/50/100. The dimensionality of the output from the transformer model ($output\_size$) is set to 20/50/100. The dimensionality of the output space of the embedding layers, as well as the input and output of the encoder and decoder layers, is set to 64. The number of attention heads ($nhead$) is 8, and the number of layers (num_layers) is 5. Similar to the LSTM model, the transformer model is trained to minimize the MSE loss function.

After obtaining the predictions from either the Transformer or the LSTM model, a three-layer Multilayer Perceptron (MLP) with a hidden size of 64 is used to map the predictions back to the length of the node feature vector. This step ensures that the predictions are compatible with the input data structure.

5.2.2. Metrics

In our study, we employed three key metrics to evaluate the performance of our models:

- **Mean Absolute Error (MAE)**: MAE is a simpler measure of error that calculates the average absolute difference between the predicted and actual values.
- **Mean Absolute Percentage Error (MAPE)**: MAPE is a normalized measure of error that calculates the average absolute percentage difference between the predicted and actual values.
- **Root Mean Squared Error (RMSE)**: This metric measures the average magnitude of the errors between the predicted values and the actual values, without considering their direction. It is calculated as the square root of the average of the squared differences between the predicted and actual values.

5.3. Sensitivity analysis

We conducted experiments on the sensitivity analysis of the learning rate parameter across different methods, examining the variation in performance metrics including MAE, MAPE, and RMSE. The experiments were designed to compare the performance of LSTM-related methods and Transformer-related methods under 20, 50 and 100 MEC servers in vehicular networks, for the task of traffic flow prediction in the next day with 7 and 14 previous days, which are denoted by 7+1 and 14+1, respectively. The range of learning rate (lr) for the LSTM-related methods and Transformer-related methods is in $[0.001, 0.01]$.

Figs. 7 and 8 present the results for sensitivity analysis of learning rate in a scenario with 20 MEC servers. The dataset consists of data from 20 MEC servers over a period of 30 days. The training and test set data are normalized using MinMax scaling. The learning rate was varied to observe its effect on the MAE, MAPE, and RMSE metrics. For each learning rate, the model will be trained 5 times with repetition, and then the average of three performance metrics will be taken as the performance representation for that learning rate. The experimental results indicate that compared to transformer-related methods, LSTM-related methods are most significantly affected by the learning rate, exhibiting greater variability. In comparison to the 7+1 prediction, the 14+1 prediction is most significantly affected by the learning rate, with greater fluctuations.

Figs. 9 and 10 present the results for sensitivity analysis of learning rate in a scenario with 100 MEC servers. The dataset consists of data from 100 MEC servers over a period of 30 days. The training and test set data are normalized using MinMax scaling. The learning rate was varied to observe its effect on the MAE, MAPE, and RMSE metrics. For each learning rate, the model will be trained 5 times with repetition, and then the average of three performance metrics will be taken as the performance representation for that learning rate. The experimental results demonstrate that, similarly, LSTM-related methods are most significantly affected by the learning rate compared to transformer-related methods, showing greater variability. When comparing the 7+1 prediction to the 14+1 prediction, the latter is most notably influenced by the learning rate, with more pronounced fluctuations.

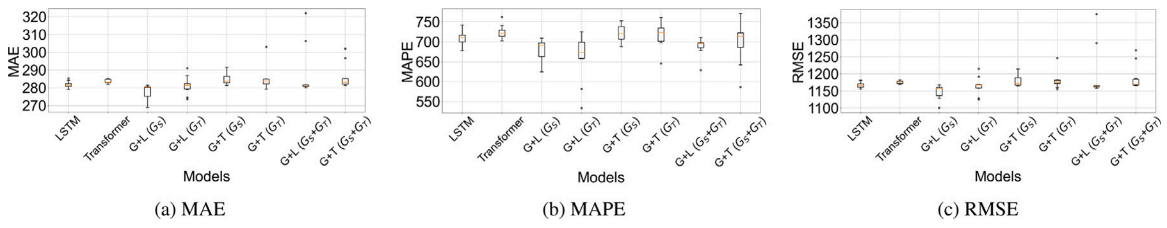


Fig. 10. Errors of the compared models under 100 MEC servers for 14+1 prediction, $lr \in [0.001, 0.01]$.

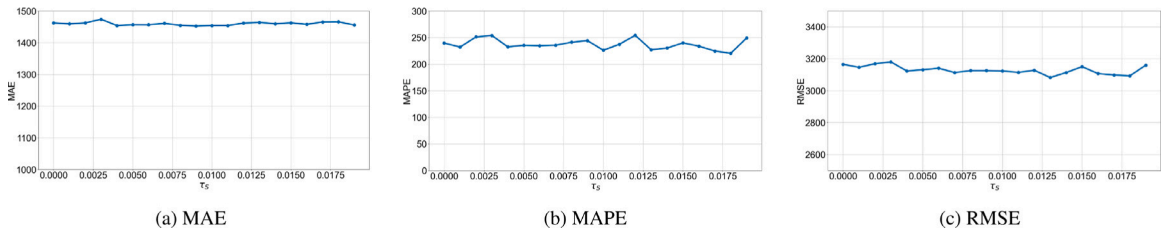


Fig. 11. Impact of threshold τ_S on errors.

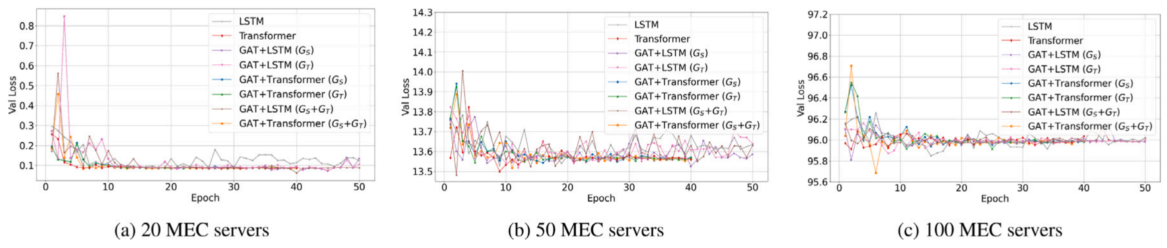


Fig. 12. Validation loss of the compared models for 7+1 prediction.

Compared to the scenario with 20 MEC servers, the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) show a downward trend in the scenario with 100 MEC servers. This is because the traffic flow is more dispersed, leading to a reduction in the absolute prediction errors for each MEC server. However, the Mean Absolute Percentage Error (MAPE) calculates the absolute error between the predicted and actual values as a percentage of the actual values, then takes the average. Moreover, MAPE is highly sensitive to situations where the actual values are zero or close to zero, as the errors are magnified. This results in a higher MAPE for the scenario with 100 MEC servers compared to that with 20 MEC servers.

Moreover, we conducted experiments regarding the threshold τ_S. Due to the scarcity of effective edges in the transition graph, τ_T is set to 0. For the similarity graph, based on the GAT+Transformer model, with all model parameters being identical and only changing the value of τ_S. We took τ_S to be in the range of [0.001, 0.02], with an interval size of 0.001. Because if the range is too large, the number of edges within the interval will decrease sharply. Fig. 11 shows the results under the metrics of MAE, MAPE and RMSE. The experimental results indicate that the threshold τ_S has a minimal impact on the model, with the values of various metrics fluctuating within a relatively small range.

5.4. Validation loss

Validation Loss (val_loss) is the loss computed on the validation set. It is used to monitor the model's performance on unseen data and to make decisions about model selection, hyperparameter tuning, and to prevent overfitting. The experimental steps for calculating the validation loss (val_loss) are as follows: For each model (predicting 1 day ahead based on the previous 7 days), training begins from epoch=1. For models related to LSTM, one model is trained for each epoch from 1 to 50, resulting in a total of 50 models. For models related to Transformer, one model is trained for each epoch from 1 to 40, resulting in a total of 40 models. The val_loss for each model is calculated using the validation set after each epoch's training.

Fig. 12 shows the results of validation loss under 20, 50 and 100 MEC servers. The experimental results indicate that transformer-related methods converge more quickly, tending to stabilize after approximately 10 epochs, whereas LSTM-related methods exhibit poorer convergence, with significant fluctuations throughout the process. On the other hand, as the number of MEC servers increases, the val loss also increases. This is because the model may exhibit signs of overfitting as the number of MEC servers increases, but it demonstrates good performance in real predictive metrics.

5.5. Ablation study

In the ablation study experiment, Table 3 shows the results of different models with 50 MEC servers for 14+1 prediction, where learning rate (lr) is 0.005. Table 4 shows the results of different models with 100 MEC servers for 7+1 prediction, where learning rate (lr) is 0.007. The experimental results indicate that the model using the Transformer performed better than the LSTM model, especially in the shorter period of historical data (i.e., 7 days). This suggests that the ability of the proposed method using GAT and Transformer models captures long-range dependencies and that its attention mechanism

Table 3
Comparison of different models with 50 MEC servers for 14+1 prediction.

Model (Graph)	RMSE	MAE	MAPE
LSTM	2123.15	714.38	891.08%
Transformer	2130.49	724.44	948.69%
GAT+LSTM (G_S)	2343.20	790.07	839.67%
GAT+LSTM (G_T)	2132.77	723.39	900.14%
GAT+Transformer (G_S)	2129.27	721.41	860.39%
GAT+Transformer (G_T)	2128.10	720.65	856.98%
GAT+LSTM (G_S+G_T)	2122.80	713.98	880.23%
GAT+Transformer (G_S+G_T)	**2122.57**	**712.74**	**825.65%**

Table 4
Comparison of different models with 100 MEC servers for 7+1 prediction.

Method (Graph)	RMSE	MAE	MAPE
LSTM	1209.45	294.28	540.35%
Transformer	1166.46	278.51	447.57%
GAT+LSTM (G_S)	1163.38	277.84	445.19%
GAT+LSTM (G_T)	1318.26	312.61	537.30%
GAT+Transformer (G_S)	1154.43	275.87	443.39%
GAT+Transformer (G_T)	1166.10	278.51	**438.13%**
GAT+LSTM (G_S+G_T)	1166.67	278.59	438.41%
GAT+Transformer (G_S+G_T)	**1147.71**	**275.05**	466.28%

may be more effective for the sequential prediction task in the context of vehicular networks with MEC servers. The results support the choice of Transformer models for traffic prediction in such environments, potentially leading to more reliable and efficient network management.

From the experimental results, it can be observed that by incorporating the GAT model on top of traditional LSTM and Transformer models, the error is reduced. This reduction in error is attributed to the establishment of associations between MEC servers through graph representation learning, thereby enhancing the accuracy of traffic flow prediction. Furthermore, after integrating the dynamic similarity graph and dynamic transition graph, the error is further reduced. This improvement is achieved by fusing multi-dimensional feature information to enhance the precision of traffic flow prediction.

6. Conclusion

For predicting edge service access volumes in 5G/6G vehicular networks, we propose the Graph Attention Network (GAT) and Transformer based model. This model integrates the power of graph-based learning with the temporal dynamics captured by the Transformer model, providing a comprehensive and accurate prediction framework for vehicular network operators. The GAT component of the model effectively captures the structural and temporal relationships between edge service nodes, allowing for a nuanced understanding of the vehicular network's topology and usage patterns. The Transformer model, on the other hand, leverages this information to predict future traffic flows, enabling proactive decision-making and efficient network utilization. Our experimental results have shown that the model achieves high accuracy in predicting edge service access volumes, with low error metrics across various datasets. This indicates that the model is robust and can generalize well to different vehicular network scenarios.

CRediT authorship contribution statement

Chao Song: Writing – original draft, Project administration. **Jie Wu:** Writing – review & editing, Supervision. **Kunyang Xian:** Validation, Software. **Jianfeng Huang:** Formal analysis, Data curation. **Li Lu:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is open access in the link declared in the manuscript: https://ieee-dataport.org/open-access/crawdad-romataxi.

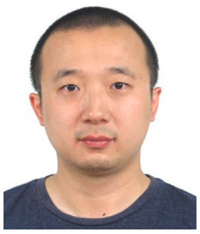
Acknowledgments

This work is supported by the National Key R&D Program of China under Grant 2021YFB3101303; the Natural Science Foundation of Sichuan Province of China No. 2024NSFSC0492; the National Natural Science Foundation of China under Grant No. 82241060, 62020106013.

References

[1] N. Aljeri, A.F.M. Boukerche, Mobility management in 5G-enabled vehicular networks, ACM Comput. Surv. 53 (2020) 1–35.
[2] H. Guo, X. Zhou, J. Liu, Y. Zhang, Vehicular intelligence in 6G: Networking, communications, and computing, Veh. Commun. 33 (2021) 100399.
[3] F. Tang, Y. Kawamoto, N. Kato, J. Liu, Future intelligent and secure vehicular network toward 6G: Machine-learning approaches, Proc. IEEE 108 (2020) 292–307.
[4] L. Liu, C. Chen, Q. Pei, S. Maharjan, Y. Zhang, Vehicular edge computing and networking: A survey, Mob. Netw. Appl. 26 (2019) 1145–1168.
[5] Y. Hui, N. Cheng, Z. Su, Y. Huang, P. Zhao, T.H. Luan, C. Li, Secure and personalized edge computing services in 6G heterogeneous vehicular networks, IEEE Internet Things J. 9 (2021) 5920–5931.
[6] B.M. Williams, L.A. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal stochastic time series process, 129, (6) 2003, pp. 664–672,
[7] Y. Xu, H. Chen, Q. Kong, X. Zhai, Y. Liu, Urban traffic flow prediction: a spatio-temporal variable selection-based approach, J. Adv. Transp. 50 (2016) 489–506.
[8] D. Yang, S. Li, Z. Peng, P. Wang, J. Wang, H. Yang, MF-CNN: Traffic flow prediction using convolutional neural network and multi-features fusion, IEICE Trans. Inf. Syst. 102-D (2019) 1526–1536.
[9] S. Hakak, T.R. Gadekallu, P.K.R. Maddikunta, S.P. Ramu, M. Parimala, C. de Alwis, M. Liyanage, Autonomous vehicles in 5G and beyond: A survey, Veh. Commun. 39 (2022) 100551.
[10] S.K. Tayyaba, H.A. Khattak, A.S. Almogren, M.A. Shah, I.U. Din, I. Alkhalifa, M. Guizani, 5G vehicular network resource management for improving radio access through machine learning, IEEE Access 8 (2020) 6792–6800.
[11] X. Cheng, R. Zhang, S. Chen, J. Li, L. Yang, H. Zhang, 5G-enabled vehicular communications and networking, Wirel. Netw. (2018).
[12] M. Noor-A.-Rahim, Z. Liu, H. Lee, M.O. Khyam, J. He, D. Pesch, K. Moessner, W. Saad, H.V. Poor, 6G for vehicle-to-everything (V2X) communications: Enabling technologies, challenges, and opportunities, Proc. IEEE 110 (2020) 712–734.
[13] W. Qi, Q. Li, Q. Song, L. Guo, A. Jamalipour, Extensive edge intelligence for future vehicular networks in 6G, IEEE Wirel. Commun. 28 (2021) 128–135.
[14] H. Zhong, L. Wang, J. Cui, J. Zhang, I.P. Bolodurina, Secure edge computing-assisted video reporting service in 5G-enabled vehicular networks, IEEE Trans. Inf. Forensics Secur. 18 (2023) 3774–3786.
[15] H.T. Nguyen, M.-T. Nguyen, H.T. Do, H.T. Hua, C.V. Nguyen, DRL-based intelligent resource allocation for diverse QoS in 5G and toward 6G vehicular networks: A comprehensive survey, Wirel. Commun. Mob. Comput. 2021 (2021) 5051328:1–5051328:21.
[16] H. Ye, L. Liang, G.Y. Li, J. Kim, L. Lu, M. Wu, Machine learning for vehicular networks: Recent advances and application examples, IEEE Veh. Technol. Mag. 13 (2017) 94–101.
[17] A. Talpur, M. Gurusamy, Machine learning for security in vehicular networks: A comprehensive survey, IEEE Commun. Surv. Tutor. 24 (2021) 346–379.
[18] A.F.M. Boukerche, J. Wang, Machine learning-based traffic prediction models for intelligent transportation systems, Comput. Networks 181 (2020) 107530.
[19] J. Prakash, L. Murali, N. Manikandan, N. Nagaprasad, K. Ramaswamy, A vehicular network based intelligent transport system for smart cities using machine learning algorithms, Sci. Rep. 14 (2024).
[20] P. Sun, N. Aljeri, A.F.M. Boukerche, Machine learning-based models for real-time traffic flow prediction in vehicular networks, IEEE Netw. 34 (2020) 178–185.
[21] H.M. Alnami, I. Mahgoub, H.A. Najada, Segment based highway traffic flow prediction in VANET using big data analysis, in: 2021 IEEE Symposium Series on Computational Intelligence, SSCI, 2021, pp. 01–08.
[22] A.F.M. Boukerche, J. Wang, Towards the design of smart vehicular traffic flow prediction, in: Proceedings of the 19th ACM International Symposium on Mobility Management and Wireless Access, 2021.
[23] J. Chen, W. Wang, K. Yu, X. Hu, M. cheng Cai, M. Guizani, Node connection strength matrix-based graph convolution network for traffic flow prediction, IEEE Trans. Veh. Technol. 72 (2023) 12063–12074.

[24] C. Song, J. Wu, W. Yang, M. Liu, I. Jawhar, N. Mohamed, Exploiting opportunities in V2V transmissions with RSU-assisted backward delivery, in: 2017 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Atlanta, GA, USA, May 1-4, 2017, 2017, pp. 271–276.
[25] C. Song, W. Yang, J. Wu, M. Liu, Red or green: Analyzing the data delivery with traffic lights in vehicular ad hoc networks, in: IEEE Global Communications Conference, GLOBECOM 2014, Austin, TX, USA, December 8-12, 2014, 2014, pp. 64–69.
[26] Wikipedia contributors, Jaccard index — Wikipedia, the free encyclopedia, 2024, https://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=1220812875. (Online; Accessed 20 May 2024).
[27] Wikipedia contributors, Kullback–Leibler divergence — Wikipedia, the free encyclopedia, 2024, URL https://en.wikipedia.org/w/index.php?title=Kullback%E2%80%93Leibler_divergence&oldid=1224220926. (Online; Accessed 20 May 2024).
[28] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: 6th International Conference on Learning Representations, ICLR, 2018.
[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS '17, 2017.
[30] D. Kang, Y. Lv, Y. Chen, Short-term traffic flow prediction with LSTM recurrent neural network, in: 20th IEEE International Conference on Intelligent Transportation Systems, ITSC 2017, Yokohama, Japan, October 16-19, 2017, IEEE, 2017, pp. 1–6.
[31] Z. Zou, P. Gao, C. Yao, City-level traffic flow prediction via LSTM networks, in: Proceedings of the 2nd International Conference on Advances in Image Processing, ICAIP 2018, Chengdu, China, June 16-18, 2018, ACM, 2018, pp. 149–153.
[32] H. Xue, F.D. Salim, TRAILER: Transformer-based time-wise long term relation modeling for citywide traffic flow prediction, 2020, CoRR abs/2011.05554. URL https://arxiv.org/abs/2011.05554.
[33] Z. Peng, X. Huang, Spatial-temporal transformer network with self-supervised learning for traffic flow prediction, in: Proceedings of the 1st International Workshop on Spatio-Temporal Reasoning and Learning (STRL 2022) Co-Located with the 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence, Vol. 3190, IJCAI 2022, ECAI 2022, Vienna, Austria, July 24, 2022, 2022.


Chao Song received his Ph.D. Degree in Computer Science from University of Electronic Science and Technology of China (UESTC), China, in 2009. During 2013, he was a visiting scholar at Temple University, under the supervision of Dr. Jie Wu. He is currently an Associate Professor in the School of Computer Science and Engineering at UESTC. His main research interests include computer networking, distributed computing and big data mining.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications.

Kunyang Xian received the B.S. Degrees in Computer Science from University of Electronic Science and Technology of China (UESTC), China, in 2023. His current research interest is in the area of graph learning.

Jianfeng Huang received the B.S. Degrees in Southwestern University of Finance and Economics, China, in 2022. His current research interest is in the area of federated learning.

Li Lu received the B.E. and M.S. degrees in automation control from Zhejiang University, Hangzhou, China, in 2000 and 2003, respectively, and the Ph.D. degree from the Key Laboratory of Information Security, Chinese Academy of Science, Beijing, China, in 2007. He is a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. His current research interests include battery-free systems and RFID technology, wireless networks, and network security.