Wiley
International Journal of Intelligent Systems
Volume 2025, Article ID 1863025, 19 pages
https://doi.org/10.1155/int/1863025

WILEY

Research Article

Energy-Efficient Resource Allocation for Urban Traffic Flow Prediction in Edge-Cloud Computing

Ahmad Ali,[1,2] **Inam Ullah**,[3,4] **Sushil Kumar Singh**,[5] **Amin Sharafian**,[1,2] **Weiwei Jiang**,[6] **Hammad I. Sherazi**,[7] **and Xiaoshan Bai**[1,2]

[1]*College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen 518060, China*
[2]*National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518060, China*
[3]*Department of Computer Engineering, Gachon University, Seongnam 13120, Republic of Korea*
[4]*Department of Artificial Intelligence, Tashkent State University of Economics, Tashkent, Uzbekistan*
[5]*Department of Computer Engineering, Marwadi University, Rajkot, Gujarat, India*
[6]*School of Information and Communication Engineering, Beijing University of Posts and Telecommunication, Beijing, China*
[7]*Department of Electrical Engineering, College of Engineering, Qassim University, Buraydah, Saudi Arabia*

Correspondence should be addressed to Xiaoshan Bai; baixiaoshan@szu.edu.cn

Received 17 October 2024; Accepted 11 January 2025

Academic Editor: Konglin Zhu

Understanding complex traffic patterns has become more challenging in the context of rapidly growing city road networks, especially with the rise of Internet of Vehicles (IoV) systems that add further dynamics to traffic flow management. This involves understanding spatial relationships and nonlinear temporal associations. Accurately predicting traffic in these scenarios, particularly for long-term sequences, is challenging due to the complexity of the data involved in smart city contexts. Traditional ways of predicting traffic flow use a single fixed graph structure based on the location. This structure does not consider possible correlations and cannot fully capture long-term temporal relationships among traffic flow data, making predictions less accurate. We propose a novel traffic prediction framework called Multi-scale Attention-Based Spatio-Temporal Graph Convolution Recurrent Network (MASTGCNet) to address this challenge. MASTGCNet records changing features of space and time by combining gated recurrent units (GRUs) and graph convolution networks (GCNs). Its design incorporates multiscale feature extraction and dual attention mechanisms, effectively capturing informative patterns at different levels of detail. Furthermore, MASTGCNet employs a resource allocation strategy within edge computing to reduce energy usage during prediction. The attention mechanism helps quickly decide which services are most important. Using this information, smart cities can assign tasks and allocate resources based on priority to ensure high-quality service. We have tested this method on two different real-world datasets and found that MASTGCNet predicts significantly better than other methods. This shows that MASTGCNet is a step forward in traffic prediction.

Keywords: attention mechanism; edge computing; internet of vehicles; resource allocation; smart city; traffic prediction

1. Introduction

Managing traffic bottlenecks in cities has become a major issue due to the growing number of cars on the road globally, further complicated by the increasing integration of IoV systems, which add new layers of data and connectivity challenges. Accurate traffic prediction is critical for drivers and Intelligent Transportation Systems (ITSs) [1, 2] to facilitate more efficient road traffic management and commuter productivity. Nevertheless, traffic forecasting is not without significant difficulties. First, traffic prediction is challenging due to the complex spatiotemporal correlations within the extensive transportation network. Second, various elements, including the climate, can cause traffic jams.

For instance, more individuals drive their automobiles rather than take public transportation when it rains. Additionally, unforeseen circumstances, like accidents or significant events, can drastically alter traffic patterns and generate forecasts that aid in more efficient traffic management. However, with data collected from various edge computing devices monitoring traffic, researchers can now use this information to understand better and predict traffic [3]. This is possible because computers are now very powerful and can quickly process information. With this new approach, scientists can build intelligent models looking at real-world data to determine how traffic works. They can better understand how traffic behaves in various situations and make predictions that help manage traffic more effectively.

Strong dynamic correlations are naturally present in both the spatiotemporal dimensions of traffic data [1]. In order to make accurate traffic predictions, it is necessary to capture these complex and nonlinear interactions between space and time, especially as IoV systems introduce additional dynamic data and connectivity, further complicating traffic flow patterns [4]. For this reason, it is still necessary to fully integrate and model these changing connections between space and time to get the best prediction models.

Traditionally, predefined adjacency graphs have employed one of two methods: (1) distance functions, which base the graph structure on the geographic separations between different nodes [5, 6]; or (2) similarity-based approaches, which provide an intuitive viewpoint by assessing node proximity by comparing the flow sequences directly, as demonstrated in Ref. [7], or by analyzing the similarity of node attributes, such as Points of Interest (PoI) data, as discussed in [8, 9]. It is crucial to remember that these methods have inherent limitations in revealing hidden spatial connections. Traffic prediction does not directly match with predefined graph structures, nor do they provide a comprehensive understanding of spatial relationships, potentially leading to significant biases [10]. Furthermore, these models have limitations in terms of generalizability beyond contexts with appropriate domain knowledge, which hinders their application to diverse domains. This, in turn, makes existing graph convolution network (GCN)-based models ineffective in scenarios beyond the boundaries of existing knowledge. The first and most challenging limitation is that extracting the simultaneous joint influences of multiple spatiotemporal dependencies is difficult.

Unfortunately, dynamically predicting spatiotemporal trends within certain areas, especially in the context of IoV systems, is a challenging topic that has not attracted much academic interest. Traffic flow is a spatiotemporal graph problem. Figure 1 displays the number of cars that enter and exit each location over a specified period. Traffic flow estimation provides insights into future traffic behavior by using past data to infer spatiotemporal trends. Regrettably, traffic prediction in multiple locations simultaneously is a significant challenge with limited research. Three major elements can impact traffic flow: the connections between locations, the effects of time on traffic, and outside influences. The inflow of region $r2$ is influenced by the outflow

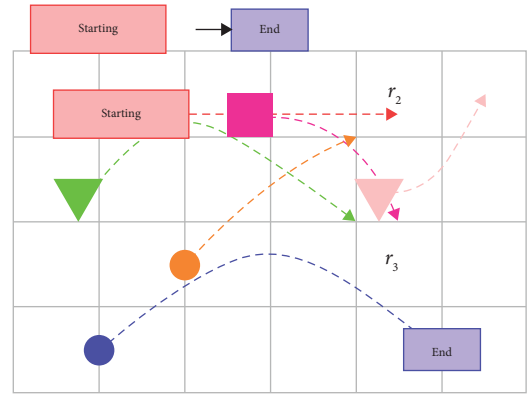

Figure 1: Traffic flows in a particular region [11].

of neighboring regions, such as $r1$ and $r3$, as well as from more distant regions, as shown in Figure 1.

Similarly, the outflow of $r2$ can influence the inflow of nearby regions. Furthermore, it is worth noting that the inflow of region $r2$ may also impact its own outflow within the same region. There are two types of spatiotemporal correlations: semantic and spatial. While the spatial relationship reflects the region's connectivity, the semantic relationship demonstrates the similarity of contextual traffic flow characteristics. During the morning rush hour, traffic flows mainly to different work areas. As a result, the work areas' spatial neighbors may also have similar traffic characteristics. Currently, the effects of spatial neighbors outweigh those of semantic neighbors. At peak times in the evening, most people may travel to business areas, while few travel to work areas. Traffic patterns in similar regions are similar.

The field of statistically motivated univariate time series prediction algorithms paved the way for the first traffic forecasting methodologies. These techniques include the vector autoregressive (VAR) [12], Historical Average (HA) [13], Seasonal Autoregressive Integrated Moving Average (SARIMA) [14], and Autoregressive Integrated Moving Average (ARIMA) [15]. A fundamental prerequisite for most of these methods is the assumption that each time series has intrinsic temporal stationary. However, it is crucial to remember that these techniques, primarily based on predefined parameters not obtained from the data itself, have inherent limitations in accurately capturing the intricate spatiotemporal correlations seen in the dataset.

The motivation for using the Multi-scale Attention-Based Spatio-Temporal Graph Convolution Recurrent Network (MASTGCNet) model is its ability to accurately predict traffic patterns over long periods in a given area, particularly, in complex environments influenced by IoV systems, where real-time data and connectivity play a crucial role in traffic dynamics. This prediction can provide valuable insights for urban planners to improve road safety, traffic management, and the overall efficiency of transportation systems. The MASTGCNet hybrid model is based on a compelling rationale that aims to competently and accurately predict long-term traffic dynamics within traffic systems. This is important for city managers to make traffic

flow smoother, keep roads safe, and make everything work better. Predicting traffic is useful for many things. It can help people plan their trips and know when to leave. Furthermore, the MASTGCNet paradigm opens new perspectives for integrating artificial intelligence methods in the ITS. However, the accurate and timely prediction of traffic flow statistics is a formidable and difficult undertaking.

Within the framework of this study, we introduce a network architecture that can generate dynamically flexible adjacency matrices. In contrast to earlier works that only dealt with static features, our suggested network uses a novel flexible generation graph framework that can adapt to the dynamic changes in traffic patterns. Our approach does not rely on standard graph structures; instead, it uses a flexible way to learn parameters for each node in the graph convolution process, considering both time and space. This integration cleverly incorporates spatiotemporal mechanisms, successfully capturing broad, long-term interdependence. At the same time, our approach incorporates a multiscale extraction feature that partitions the input characteristics into four distinct parallel partitions of different dimensions. Even with MASTGCNet encouraging outcomes, several obstacles still exist to overcome in practical implementations. First, the quality and availability of real-time traffic data, which might vary greatly between cities, are critical to the model's effectiveness. Second, the scalability of edge-cloud architecture might be limited in expansive urban settings with diverse infrastructure. Finally, a significant obstacle to deployment in smart city systems is striking the ideal balance between prediction accuracy and energy efficiency during times of high demand. This paper highlights the following main contributions:

- We propose a MASTGCNet. This network can carefully include data from changing graphs. This model combines spatiotemporal extraction feature mechanisms at several scales, which makes it better at processing information from a wide range of receptive regions at various hierarchical levels. Furthermore, MASTGCNet employs a resource allocation strategy within edge computing to reduce energy usage during the prediction process. The attention mechanism helps quickly determine which services are most important.
- We introduce a new module for extracting multiscale spatiotemporal features. This module divides input characteristics into four parallel partitions of varying dimensions, improving the capture of complex contextual information and resilience to scale changes. It also has two types of attention mechanisms that work together to understand better how space and time are connected, improving the ability to find correlations on various scales.
- We demonstrate the effectiveness of our proposed model through extensive experiments conducted on two different real-world traffic datasets. Our model is superior to the prevailing approaches in these empirical evaluations, demonstrating its effectiveness in traffic flow prediction.

We organize the remainder of this paper as follows: In Section 2, we expose the problem formulation. Our work methodology is briefly explained in Section 3. Section 4 describes the system architecture. Then, in Section 5, we discuss the extensive experiments conducted on two real-world traffic datasets, including empirical investigations. In Section 6, we provide the related literature. Finally, we present our concluding remarks in Section 7.

2. Problem Formulation

In this section, we formally introduce the traffic prediction problem and mathematically describe a traffic network's concepts. Table 1 shows the summary of the most important notations.

The task of predicting traffic flow can thus be viewed as a complex time series multiscale prediction problem, enhanced by incorporating auxiliary prior knowledge. In general, this involves predicting a weighted graph, typically denoted as $|G| = (S, E, A)$, where S signifies the nodes set indicating the sources of the traffic flow and $|S| = N$ denotes the number of nodes. At the same time, $E = (i, j, w_{ij})$ is a set of edges, the edge weight w_{ij} among nodes i and j represents distance, travel time, and other relevant factors. Additionally, $A \in R^{N\times N\times T}$ represents a spatial matrix that characterizes the internode similarities, encompassing factors, such as the distance of road network and PoI similarity. This matrix is pivotal as foundational information input for the graph convolution process. More specifically, we consider a collection of traffic data consisting of N related univariate time series denoted as $Y = (X{:}\ 0, X{:}\ 1, \ldots, X{:}\ t, \ldots)$ where each constituent series $X{:}\ t = (x1, t, x2, t, \ldots, x_i, t, \ldots, X_N, t)$ is encapsulated in a vector $S \in R^{N\times 1}$ that enumerates the compilation of N sources at the discrete time instance t. Our goal is to predict upcoming values within these traffic patterns using information from historical observations.

The objective of traffic forecasting is to learn a function, denoted as F, with the capability to predict a future signal tensor θ based on a set of historical signal tensors T and the underlying graph structure G. This function predicts subsequent data spanning time steps τ ahead, and k represents the time step,

$$[X_{k+1}, \ldots, X_{k+\tau}] = H_\phi(X_{k-T+1}, \ldots, X_k; G), \tag{1}$$

$$X_{m+k} = H_\phi(X_{m-T+1}, \ldots, X_m; G)_k, \quad \text{for } k = 1, 2, \ldots, \tau. \tag{2}$$

Table 1: Summary of major notations.

Notations	Description
G	Weighted graph
E	Set of edges
S	Set of nodes
$Y: t, b_t$	Input and output at time t
$A \in R^{N \times N}$	Adjacency matrix
$b \in R^F$	Biases
W_n	Reservoir weights
N_n	Node matrix
Q	Query subspace
V	Value subspace
K	Key subspace
$X(t)$	Predicted traffic flow
R_i	Resource allocation

3. Methodology

The proposed MASTGCNet model is shown in Figure 2. In this section, we first present the structure of our proposed MASTGCNet model. We then provide a detailed explanation of each component within this model.

3.1. Model Framework. The attention layer, encoder layer, and prediction layer are the three primary layers that comprise the MASTGCNet model framework. The traffic data are initially converted using a tanh activation function for stability, starting with the flow input. A feature embedding block is then used to capture complicated patterns. After that, the traffic data are analyzed through several spatiotemporal attention (ST-Attention) layers to identify geographical and temporal correlations. The encoded data are sent to the encoder layer, which manages temporal patterns and dynamic traffic variations using multiscale temporal unit (MUST) modules and Dynamic Graph Generation Networks (DGGNs). After the traffic flows are adjusted, they enter the prediction layer, where a GCN is used to capture spatial dependencies, and a sequence of ResUnits further develop deep representations. Ultimately, the prediction output is generated by a fully connected (FC) layer, which uses the processed data to represent the expected traffic flows.

Complex spatiotemporal relationships are required for traffic prediction. As shown in Figure 2, our study proposed the MASTGCNet model. This network combines the principles of the DGGN, multiscale attention, and gated recurrent units (GRUs). The intention is to effectively capture both spatiotemporal relationships between nodes within traffic flow sequences. In the MASTGCNet model, the conventional MLP layer within the GRU is replaced by the DGGN, which enables the detection of node-specific patterns. In addition, the DGGN module inherently identifies spatial dependencies, contributing to a comprehensive understanding of the underlying traffic dynamics. The MASTGCNet formulation is as follows:

$$s_i = \sigma\left(\widehat{A}\left[X_i: t, b_t - 1\right] \cdot W_s + Nb_s\right), \tag{3}$$

$$t_i = \sigma\left(\widehat{B}\left[X_i: t, b_t - 1\right] \cdot W_t + Nb_t\right), \tag{4}$$

$$b_n = \text{softmax}\left(\sigma A\left[X: t, r \odot b_t - 1\right] N W_{\overline{b}} + Nb_{\overline{b}}\right), \tag{5}$$

$$b_t = \text{STMU}\left[s \odot b_t - 1 + (1 - s) \odot \overline{b}\right], \tag{6}$$

where $X: t$ and b_t represent the input and output at given time t, respectively. $\odot$ denotes the concatenation operation, while n and s symbolize the update and reset gates. Each node undergoes an initial random assignment using a learnable node embedding dictionary denoted by N. The parameters NN, W_s, W_n, W_b, b_s, b_n, and b_b are considered trainable entities in the MASTGCNet framework. Like the GRU architecture, all of these parameters can undergo end-to-end training within MASTGCNet via backpropagation over time. By examining equation (3), it becomes clear that MASTGCNet integrates all embedding matrices as a singular entity denoted by N.

3.2. Modeling Dynamic Generation Graph Network Module. Urban traffic flow conditions, especially with integrating IoV systems, frequently undergo rapid and significant fluctuations within short intervals, adding complexity to real-time traffic management. The underlying topological structure of the input adjacency matrix can influence the effective feature extraction by the GCN. To address this issue and flexibly respond to input node information, we present a novel framework that combines the GCN with GRU. This hybrid model improves prediction accuracy while minimizing parameter usage and has a simpler structure than recurrent neural networks (RNNs). The model employs stacked fusion graph convolution modules to capture multilevel spatial correlations and enhanced GRUs to extract multiscale temporal correlations, effectively extracting the complex spatiotemporal correlation between traffic flow data. By incorporating graph generation for each node, the model dynamically captures the finer details of the road network topology, thus achieving higher granularity in its representation. The mathematical formulations for the GCN and GRU are as follows:

$$Z(t) = \text{GCN}(X, A, H(t-1)), \tag{7}$$

$$H(t), X(t) = \text{GRU}(Z(t), H(t-1)), \tag{8}$$

where $Z(t)$ denotes the hidden state of the model at time step t, $H(t-1)$ denotes the hidden state of the model at the previous time step $(t-1)$, X denotes the input features, while $H(t)$ is the updated hidden at time step t.

Within graph convolution operations, it is common practice to share parameters and biases (b) uniformly across all nodes. However, such parameter sharing can potentially limit the model's ability to extract complex features

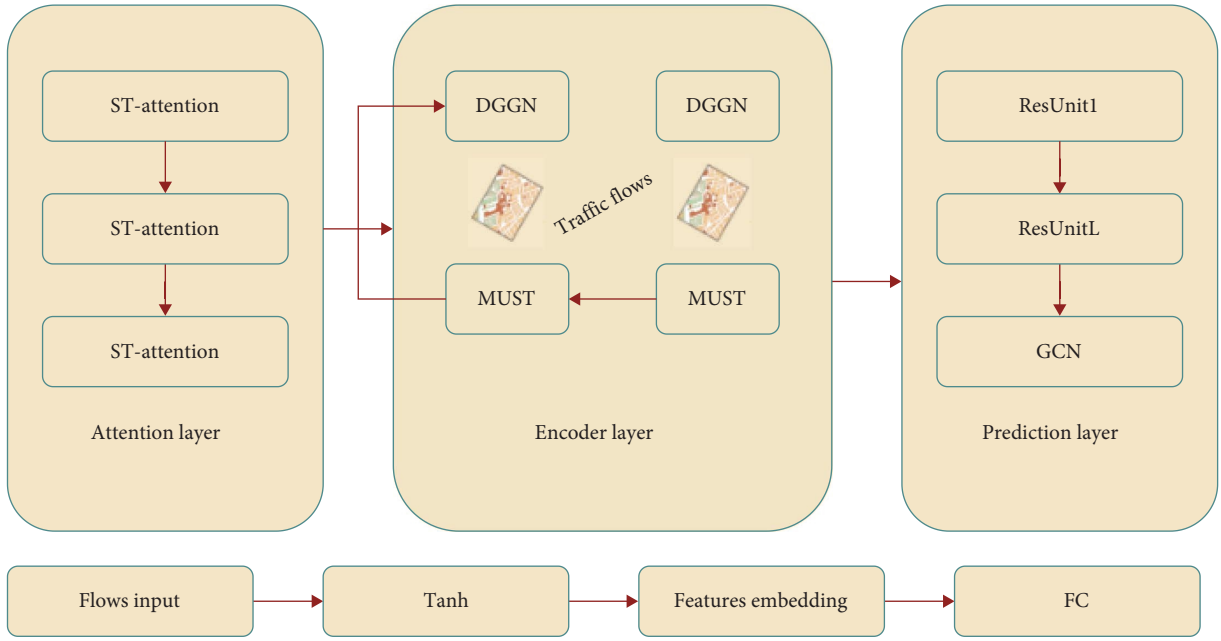

Figure 2: Framework of the MASTGCNet model comprises prediction, attention, and encoder layers.

effectively. This observation is relevant to traffic scenarios, where neighboring nodes may manifest distinct traffic patterns due to their unique attributes. Furthermore, we can observe different flow sequences, characterized by opposite patterns, between two nonoverlapping nodes. These considerations highlight the need to identify specific patterns for each node, necessitating the maintenance of a distinct parameter space for each node. In Ref. [16], it was shown that the first-order Chebyshev polynomial expansion can be used to get close to the graph convolution operation,

$$Z = b + Y\theta\left(1 + D^{(-1/2)}AD^{(-1/2)}\right), \tag{9}$$

$$G_{\text{Conv}} \approx T_1(L) \cdot Y \cdot \ominus. \tag{10}$$

Consider a graph denoted by its adjacency matrix $A \in R^{N\times N}$ and the associated degree matrix D. Additionally, let $\theta \in R^{C\times F}$ and $b \in R^F$ symbolize the learning weights and biases, respectively. Here, X denotes the input, while Z denotes the output derived from the GCN layer. In the context of isolating a single node i for analysis, the operation performed by the GCN can be construed as effecting a transformation of the features of the node from an initial state, represented as $X^i \in R^{1\times C}$, to a result in the state denoted as $Z^i \in R^{1\times F}$. Note that G_{Conv} represents the graph convolution operation, $T_1(L)$ denotes the first-order Chebyshev polynomial applied to the graph Laplacian matrix L, and X represents the input feature matrix. In contrast, $\ominus$ denotes the learnable weight parameters.

The DGGN introduces an automated mechanism that dynamically generates the graphs and features in response to the input data encountered at every successive time step. This adaptive approach allows for the better capture of hidden dependencies between nodes. In the DGGN module, individual nodes are given a randomized initialization via a trainable node embedding dictionary denoted as $N \in R^{N\times D}$, where a dedicated row denotes every embedding node's profile in N. Here, D denotes the embedding vertex dimensionality. The establishment of graph similarity relies on the node affinities, and spatial correlations among node pairs are inferred by the matrix product of N with its transpose, N^T. Throughout the training program, N is subject to automatic updates, enabling the discovery of concealed dependencies among various flow sequences. The result is an adaptive matrix tailored to graph convolutions. Importantly, the adaptive matrix undergoes a normalization process, enhancing its utility and significance,

$$D^{(-1/2)}\text{AD}^{(-1/2)} = \tanh\left(\text{ReLU}\left(N \times N^T\right)\right). \tag{11}$$

The DGGN uses the model defined in equation (11) to generate the generalization of the term $D^{(-1/2)}\text{AD}^{(-1/2)}$ to the high-dimensional context of the GCN. Hence, mitigating the need for repetitive computations throughout the iterative training process is expressed mathematically in equation (14),

$$Z = b + Y\ominus\left(I + \tanh\left(\text{ReLU}\left(N \times N^T\right)\right)\right), \tag{12}$$

$$M = I + \tanh\left(\text{ReLU}\left(N \times N^T\right)\right), \tag{13}$$

$$Z = b + Y \ominus M. \tag{14}$$

Assigning different parameters to individual nodes leads to a large number of parameters, leading to challenges in optimizing these parameters in subsequent training stages, culminating in overfitting concerns. To address this issue, the DGGN method strategically decomposes the parameters, denoted as $\ominus_1, 2, \ldots, N \in R^{N\times C\times F}$, assigned to each node. This decomposition effectively splits the initial parameters

into two comparatively smaller parameter matrices as given by

$$\Theta_k = W_n \times N_n, \quad \text{for } k = 1, 2, \ldots, N. \tag{15}$$

The above equation satisfies both a reservoir of weights represented by $W_n \in R^{d\times C\times F}$ and a matrix of node embeddings denoted as $N_n \in R^{N\times d}$, where d denotes the embedding dimension, which is notably less than N. By substituting the previously articulated parameter set $\Theta_k, \ldots, N \in R^{N\times C\times F}$ with the matrix multiplication product of these two parameter matrices, a noticeable reduction in the number of parameters is achieved. This methodology also extends to the treatment of bias terms. Finally, the advanced GCN integrated within the DGGN framework can be formulated as follows:

$$Z = b_g N_g + XW_n N_n\left(I + \tanh\left(\mathrm{ReLU}\left(N \times N^T\right)\right)\right). \tag{16}$$

Given an individual node, denoted as i, the procedure involves extracting parameters Θ_i, unique to the i node, with a widely shared reservoir weight W_n, while exploiting the inherent node embeddings N_n^i. Conceptually, this process can be seen as a mechanism for acquiring node-specific patterns by identifying specific patterns inherent in a broader collection of prospective patterns derived from all the traffic sequences.

3.3. Modeling Dynamic Multiscale Spatiotemporal Feature Unit. Adjacent road segments often exhibit traffic flow characteristics where the predicted state of traffic at a given node depends on the traffic dynamics of its neighbors. This outcome is predicted by a confluence of factors, including observed time intervals, immediate historical context, and abrupt variations such as traffic incidents and meteorological conditions. However, individual nodes exert a highly variable degree of influence. In response to the challenge posed by the inability of graph convolutions to allocate varying weights to individual nodes, we have designed a MUST. We designed this module to encapsulate both spatiotemporal correlations within traffic nodes, thereby improving the accuracy of node-level traffic flow predictions. The MUST module consists of three main parts: temporal attention, spatial attention, and a multiscale module. The temporal and spatial modules represent the input as a tensor of three dimensions, encapsulating F-dimensional features about N nodes over successive time instants $(Y{:}\ t, Y{:}\ t-1, \ldots, Y{:}\ t-T+1)$. We performed this extraction using the previously introduced DGGN. The mathematical formulation for long-term temporal correlations is as follows:

$$X_{l:t}, X_{l:t-1}, \ldots, X_{l:t-T_l+1}, \tag{17}$$

where l represents the long-term temporal scale, while X_l denotes the features at the long-term scale.

3.4. Modeling Attention-Based Spatiotemporal Module. Figure 3 depicts the temporal transformer, denoted as TT, a novel construct introduced in this work to encapsulate persistent long-term temporal dependencies effectively. In contrast to alternative neural network frameworks, the TT demonstrates an enhanced ability to accommodate long-term temporal dependencies seamlessly. Similarly, the module of the spatial transformer, labeled ST, adopts a design parallel to that of the TT, further underscoring its utility in addressing spatial transformations. To initiate the procedure, the TT starts with a 1×1 convolutional layer to process the input features,

$$X' = \mathrm{Conv}_{1\times 1}(Z), \tag{18}$$

$$X' = W \cdot Z, \tag{19}$$

where $Z \in R^{T\times N\times F}$, the operation of $\mathrm{Conv}_{1\times 1}$ produces vectors of d_g dimensions for each node over temporal intervals, while W is the weight matrix for the 1×1 convolution. This convolution process is further characterized by using concurrent processing across nodes to capture temporal dependencies effectively. At the same time, we introduce the two-dimensional spatial feature tensor, denoted as $\overline{X} \in R^{T\times d_g}$, which represents the features associated with each node within the graph G. The time series $\overline{X} \in R^{T\times d_g}$, as input for the temporal attention unit, we employ a sliding window with a length of T and a channel size of d_g. By applying dynamic computations within high-dimensional latent subspaces, like query subspace $Q \in R^{T\times d_g}$, the key subspace $K \in R^{T\times d_A}$, and the value subspace $V \in R^{T\times d_g}$, temporal dependencies are extracted and delineated,

$$Q = \overline{Y} \cdot W_q, \tag{20}$$

$$K = \overline{Y} \cdot W_k, \tag{21}$$

$$V = \overline{Y} \cdot W_v, \tag{22}$$

where $W_q \in R^{d_g\times d_A}$, $W_k \in R^{d_g\times d_A}$, and $W_v \in R^{d_g\times d_g}$ denotes the learned linear mappings. Using data from the past T time steps, we predict the future values for the next T time steps. For example, given data at times $(X{:}\ t, X{:}\ t-1, \ldots, X{:}\ t-T-1)$ predict $(Y{:}\ t, Y{:}\ t+1, Y{:}\ t+2, \ldots, Y{:}\ t+\tau)$. To make predictions for multiple future steps, we use a method called a scaled dot product function. This helps us understand how temporal patterns work in both directions in historical traffic data. To make our predictions even more accurate, we combine another set of information, represented as V, with time-related details, represented as W. This helps us understand how different factors are related over time. We also use a FC neural network. This system helps us explore the relationships between hidden features, making our predictions better and more reliable,

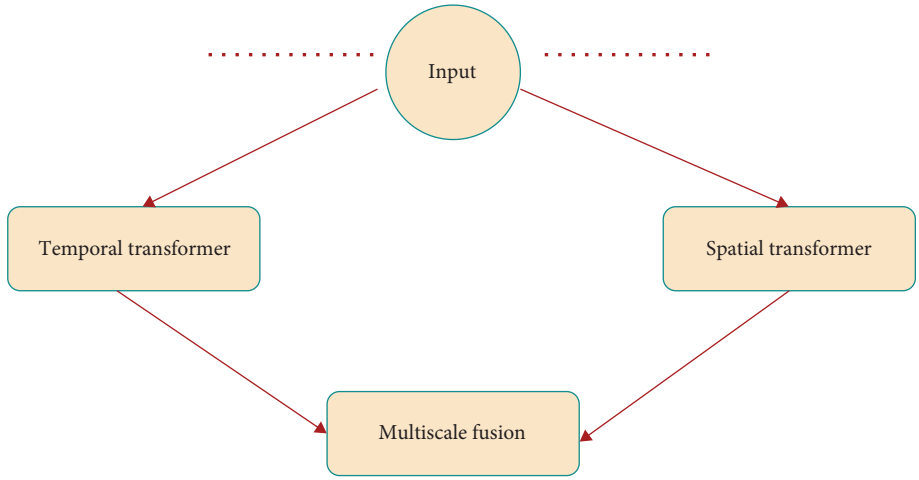

Figure 3: MSUT consists of spatial and temporal transformers with multiscale features.

$$W = \tanh\left(\frac{Q \cdot K^T}{\sqrt{d_A}}\right), \tag{23}$$

$$S = M \times V, \tag{24}$$

$$U = f(S), \tag{25}$$

where K^T and Q represent the matrix product and normalized to achieve the attention distribution in each time step, while $\sqrt{d_A}$ denotes the weight scaled factor and the function of tanh is employed to normalize the weight score.

To ensure robustness, we use a residual connection $W' = W + Z$ during training phase. The resulting output of every node is denoted as $Y' = U + W'$, and this process ultimately leads to the parallelization of outputs across all nodes, yielding $X \in R^{T \times d_g}$.

To extend the scope of time series forecasting for extended prediction horizons, we intelligently capture bidirectional temporal dependencies over an extended period. This is accomplished within the structure of a sliding window, operating at each discrete temporal instance. The hierarchical architecture of the TT effectively captures complex dependencies across multiple layers, thereby enabling the prediction of long sequences without sacrificing computational efficiency by increasing the parameter T. In contrast, RNN-based methods face vanishing gradients, while models based on convolutional methods require an explicit specification of an expanding convolutional layer concerning the parameter T.

3.5. Multiscale Feature Extraction Unit. In the domain of traffic flow prediction, the dynamics of nodes are subject not only to the influence of close neighbors but also to the states prevailing in distant geographical regions. The spatiotemporal multiscale feature extraction module creates a multiscale feature extraction process that prioritizes spatial characteristics to enhance the extraction of nodes' intrinsic spatial characteristics. Multiscale convolution kernels facilitate this effort by exploiting the extraction of different spatial resolutions and depths. This approach can capture a wider range of informative positional cues embedded in the input tensor. The resulting multiscale feature representation lends to parallel processing strategies and promotes enhanced contextual insight. However, the multiscale convolution kernel's dimensional expansion leads to a proportional increase in the number of parameters. The adopted strategy incorporates the group convolution technique to prevent increased computational complexity. In this approach, the input tensor, or equivalently, the node matrix, is partitioned into a large number of groups. In particular, the nodes within every distinct group collectively utilize a uniform convolution kernel,

$$H(t), X(t) = \mathrm{GRU}\left(\left(\mathrm{GCN}(X, A) \odot A1(t)\right) \odot A2(t), H(t-1)\right), \tag{26}$$

where $X(t)$ represents the predicted traffic flow at time step t, while $H(t)$ represents the hidden state at time step t.

4. System Architecture

In Figure 4, we illustrate the architecture of our system, which comprises three primary components: local GPU servers, the Cloud, and users (e.g., through websites). This setup facilitates both offline and online data flows. The local GPU servers store historical observations, such as taxi trajectories and meteorological data. The Cloud receives real-time data, including real-time traffic information (such as trajectories within specific intervals) and meteorological updates. The suggested energy-efficient resource allocation plan for the edge-cloud architecture is the main topic of Section 4. The attention mechanism determines the importance of prediction tasks, and the method dynamically allocates computational resources accordingly. This enables the system to optimize energy consumption by prioritizing high-demand operations, such as busy traffic regions or crucial crossings, while allocating fewer resources to less crucial jobs. While the cloud handles more complicated activities, the edge servers manage time-sensitive predictions

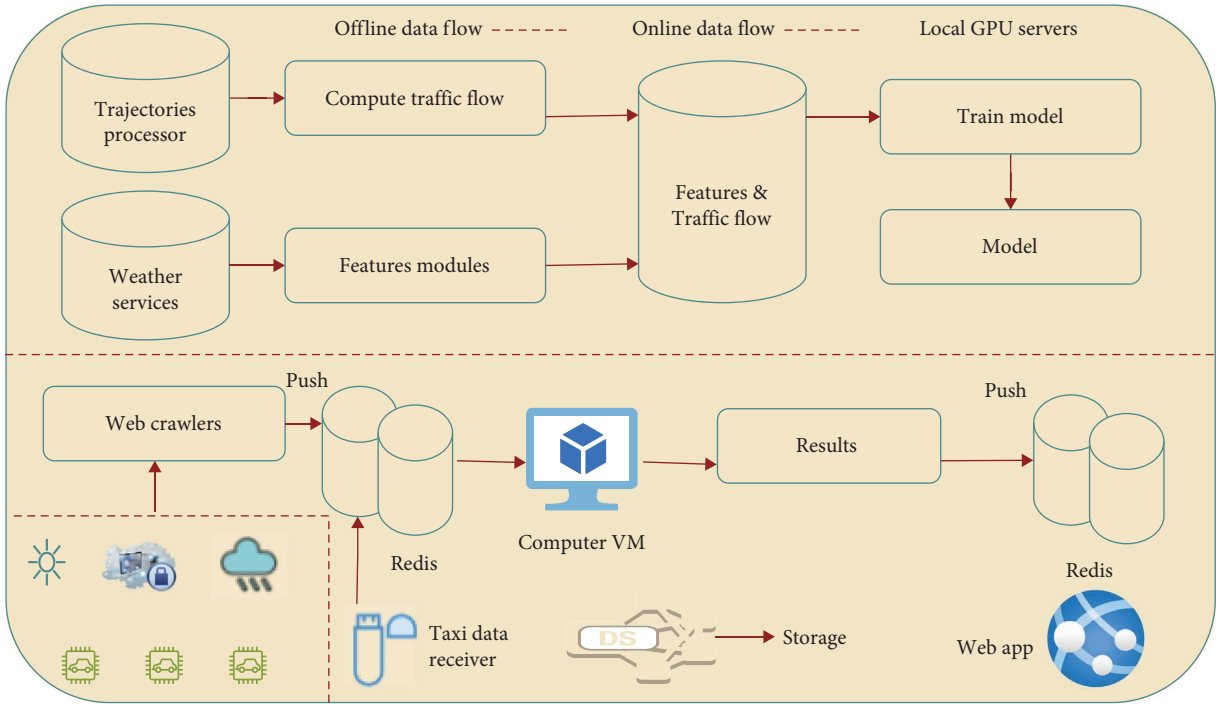

Figure 4: System architecture.

to reduce latency. The precise resource allocation algorithm, the metrics used to assess resource efficiency, and the experimental findings that show how well this strategy works to lower energy consumption while preserving high prediction accuracy are just a few of the extra details we will cover in this section. Users interact with the system to access both inflow and outflow data.

4.1. Efficient Cloud-Based Traffic Flow Analysis. The edge computing system continuously receives GPS trajectories of taxicabs and aggregates meteorological data, demonstrating a commitment to energy-efficient resource allocation and task assignment. These data streams are efficiently stored in a Redis cache, minimizing storage overhead and energy consumption. Leveraging virtual machine (VM) instances or multiple VMs on the Cloud, we employ energy-efficient resource allocation strategies to retrieve and process data from the Redis cache. By computing crowd flows for each city region based on GPS trajectories, we ensure optimal utilization of computational resources while minimizing energy consumption. Furthermore, our VM instances implement energy-efficient task assignment protocols, extracting features from meteorological data and other event data with precision and efficiency [17, 18]. To optimize resource usage on the cloud and mitigate costs associated with additional storage, we judiciously retain only crowd flow data and features from the past 2 days. Historical data are periodically transferred to local servers, ensuring data accessibility while minimizing energy consumption on the cloud. The mathematical formulation for our system is as follows:

$$\min_{\mathscr{R}} \ \mathscr{E}(\mathscr{R}), \tag{27}$$

$$T_{\text{proc}}(\mathscr{D}_t, \mathscr{D}_w) \le T_{\max}, \tag{28}$$

$$\text{Accuracy}(\mathscr{M}(\mathscr{F}_t, \mathscr{F}_w)) \ge P_{\min}, \tag{29}$$

$$\sum_{i=1}^{N} \mathscr{R}_i \le \mathscr{R}_{\max}, \tag{30}$$

$$\mathscr{R}_i = f(\mathscr{D}_t, \mathscr{D}_w, \mathscr{M}), \tag{31}$$

$$\mathscr{P} = g(\mathscr{F}_t, \mathscr{F}_w, \mathscr{M}), \tag{32}$$

where $T_{\text{proc}}(\mathscr{D}_t, \mathscr{D}_w)$ denotes the processing time for traffic and weather data, $\text{Accuracy}(\mathscr{M}(\mathscr{F}_t, \mathscr{F}_w))$ represents the prediction accuracy of the model, $\sum_{i=1}^{N} \mathscr{R}_i$ denotes the total allocated resources, while $\mathscr{R}_i = f(\mathscr{D}_t, \mathscr{D}_w, \mathscr{M})$ is the resource allocation function.

In the context of edge-cloud computing, the prediction model performance is optimized in large part by the energy-efficient resource allocation. Through the distribution of computational activities between edge servers and the cloud, the edge-cloud architecture guarantees low latency and energy-efficient processing. By prioritizing tasks using the attention mechanism, the resource allocation technique

enables the system to distribute computational resources and reduce energy usage dynamically. This method guarantees timely and effective traffic projections, particularly in smart city settings. This dual focus is reflected in the title, highlighting energy efficiency and traffic prediction, ensuring that both are well represented in the final solution.

We leverage the Azure platform as a service (PaaS) for our system, ensuring efficient resource allocation and task assignment, particularly when processing the vast amounts of data generated by IoV systems. Table 2 offers a comprehensive breakdown of the Azure resources utilized in our setup, meticulously optimizing for both performance and cost-effectiveness. In forecasting crowd flows for the near future, we prioritize energy-efficient resource allocation by employing a VM known as the A2 standard in Azure. With its 2 cores and 3.5 GB memory, this VM strikes a balance between computational power and energy consumption. To efficiently handle potential heavy traffic from numerous users, especially the data influx from IoV systems, we employ an energy-efficient task assignment strategy by hosting both the website and web service on an App Service. Furthermore, to minimize energy consumption while ensuring data accessibility, historical data are stored on local servers, while a 6-GB Redis cache efficiently caches real-time trajectories from the past half-hour, crowd flow data, extracted features from the past 2 days, and inferred results.

4.2. Local GPU Servers. While many tasks can be performed on the cloud, GPU services are unavailable in certain regions such as China. Implementing energy-efficient resource allocation and task assignment strategies is essential in optimizing cloud-based operations. Additionally, there are expenses associated with other edge computing services, such as storage and I/O bandwidth. Therefore, cost-saving measures are crucial for a research prototype. Moreover, transferring large volumes of data from local servers to the cloud is time-consuming due to limited network bandwidth. Energy-efficient data transfer protocols and task prioritization algorithms can help mitigate delays caused by data migration. For example, historical trajectories can span hundreds of gigabytes or even terabytes, resulting in significant delays when copying data from local servers to the cloud. Efficient compression algorithms and prioritization mechanisms can be employed to streamline the transfer process and minimize energy consumption.

Hence, we adopt a hybrid framework that merges local GPU servers with cloud resources. The primary role of local GPU servers is to manage offline training processes, encompassing three main tasks:

- Transforming trajectories into inflow/outflow data involves utilizing extensive historical trajectory datasets and employing a computation module to derive crowd flow information, which is subsequently stored locally.
- Gathering features from external sources entails initially aggregating external datasets from various data repositories, such as weather and holiday events. These datasets are then processed through a feature extraction module to generate continuous or discrete features stored locally for further analysis and utilization.
- To train our predictive model, we utilize the generated crowd flows and external features through our proposed MASTGCNet framework. The trained model is then uploaded to the cloud. Since the dynamic crowd flows and features are stored in Azure Storage, we synchronize the online data with our local servers before each training session. This approach allows us to experiment with new ideas, such as retraining the model, while significantly reducing costs for a research prototype.

Table 2: Azure services and prices.

Azure service	Configuration	Prices
Virtual machine	Standard A2 (2 cores, 3.5 GB memory)	$0.120/h
App service	Shared plan, A2 standard, 2 cores	$0.400/h
Redis cache	6 GB, PI premium	$0.55/h

4.3. Training Process. Algorithm 1 describes the training procedure of MASTGCNet. Backpropagation is used randomly to initialize and optimize the trainable parameters based on MASTGCNet. Backpropagation minimizes the loss function of our proposed model using the stochastic gradient method. We implement the dropout strategy approach to increase the overall effectiveness of our method. Algorithm 2 aims to optimize resource allocation for energy-efficient traffic flow prediction. During training, it iteratively adjusts the allocation of computational resources based on the processing time, prediction accuracy, and energy consumption. The optimization process involves fine-tuning the model and resource distribution to meet the constraints of maximum allowable processing time; minimum required prediction accuracy, and maximum available resources, ensuring that the model achieves high accuracy with minimal energy usage.

5. Experiments and Performance Analysis

We have implemented and empirically evaluated the MASTGCNet model using two large traffic datasets, BikeNYC and TaxiBJ. We first talked about the datasets we used, how we set some parameters, and how we measured how well the model worked. Next, we discussed other models recently proposed for traffic prediction.

5.1. Experiment Setup and Datasets. We use a Linux server with numerous software, as defined later in Section 5.3, and hardware described in Table 3.

In this experimental study, we focus on predicting traffic flows within two large datasets: BikeNYC and TaxikBJ. Detailed information about these datasets is explicitly provided in Table 4.

Input: Historical observations: $\{X_t | t = 1, 2, \ldots, T\}$; The spatial graph Z: $\{N_t | t = 1, 2, \ldots, n\}$; Pre-defined graph $\{|G| = (S, E, A)\}$;
Output: Learned MASTGCNet Model

1. $D \longleftarrow 0$;
2. **for** all accessible time interval t $(2 \leq t \leq n)$ **do**
3. $Z = [b + X(1 + D^{(-1/2)}AD^{(-1/2)})]$;
4. $D^{(-1/2)}AD^{(-1/2)} = \tanh(\mathrm{ReLU}(\mathrm{NN}^T))$;
5. $Z = b + X\ominus(I + \tanh(\mathrm{ReLU}(\mathrm{NN}^T)))$;
6. Put a training instance (X_t) into D;
7. **end for**
8. // Training the model;
9. initialize learnable parameters $\ominus$ in MASTGCNet;
10. **repeat**
11. randomly chose batch of instances D_{batch} from D;
12. calculate $\ominus$ by reducing the objective with D_b;
13. **until** *model criteria met*;
14. **return** *MASTGCNet model*

Algorithm 1: The MASTGCNet algorithm.

Input: Traffic data $\mathscr{D}_t$, Weather data $\mathscr{D}_w$, Maximum allowable processing time $T_{\max}$, Minimum required prediction accuracy $P_{\min}$, Maximum available resources $\mathscr{R}_{\max}$
Output: Optimized resource allocation $\mathscr{R}$, Predicted traffic flow $\mathscr{P}$

1. Initialize resource allocation $\mathscr{R} \longleftarrow \{\mathscr{R}_i \mid i = 1, 2, \ldots, N\}$;
2. Initialize model $\mathscr{M}$;
3. **while** not converged **do**
4. Compute traffic features $\mathscr{F}_t \longleftarrow$ Compute Features $(\mathscr{D}_t)$;
5. Compute weather features $\mathscr{F}_w \longleftarrow$ Compute Features $(\mathscr{D}_w)$;
6. Compute processing time $T_{\mathrm{proc}} \longleftarrow$ Compute Processing Time $(\mathscr{D}_t, \mathscr{D}_w)$;
7. Compute prediction accuracy Accuracy $\longleftarrow$ Evaluate Model $(\mathscr{M}, \mathscr{F}_t, \mathscr{F}_w)$;
8. **if** $T_{\mathrm{proc}} > T_{\max}$ **then**
9. Adjust resources to reduce processing time: $\mathscr{R} \longleftarrow$ Adjust Resources $(\mathscr{R}, T_{\max})$;
10. **end if**
11. **if** Accuracy $< P_{\min}$ **then**
12. Adjust model or resources to improve accuracy: $\mathscr{M}, \mathscr{R} \longleftarrow$ Improve Accuracy $(\mathscr{M}, \mathscr{R}, P_{\min})$;
13. **end if**
14. Update total energy consumption $\mathscr{E}(\mathscr{R}) \longleftarrow \sum_{i=1}^{N}(P_i \cdot T_i)$;
15. **if** $\mathscr{E}(\mathscr{R}) > \mathscr{R}_{\max}$ **then**
16. Adjust resources to minimize energy consumption: $\mathscr{R} \longleftarrow$ Minimize Energy $(\mathscr{R}, \mathscr{R}_{\max})$;
17. **end if**
18. **end while**
19. Compute final predicted traffic flow $\mathscr{P} \longleftarrow \mathscr{M}(\mathscr{F}_t, \mathscr{F}_w)$;
20. **return** $\mathscr{R}$, $\mathscr{P}$.

Algorithm 2: Energy-efficient resource allocation for traffic flow prediction.

Table 3: System specifications.

Component	Specification
CPU	8 Intel (R) Core (TM) Xeon (R) CPU E5-2680; v4 @3.80 GHz
GPUs	4 NVIDIA P100
cuDNN version	8.0
CUDA version	8.0
RAM	256 GB

- TaxiBJ: The city of Beijing provided the taxi trajectory data for this dataset over 16 months: from July 1, 2013, to October 30, 2013, from March 1, 2014, to June 30, 2014, from March 1, 2015, to June 30, 2015, and from November 1, 2015, to April 10, 2016. The dataset contains over 35,000+ taxi trajectories, effectively

Table 4: Traffic dataset information.

Datasets	TaxiBJ	BikeNYC
Urban	Beijing	New York
Format of data	Taxi GPS	Rent bike
Time span	July 1, 2013–October 30, 2013 March 1, 2014–June 30, 2014 March 1, 2015–June 30, 2015 November 1, 2015–April 10, 2016	February 5, 2014–August 29, 2014
Period	30 min	1 h
Taxis and bikes	35,000+ taxis	6900+ bikes
Map size	32×32	16×8

capturing urban traffic flows in Beijing. We partition the data, using a subset for training and reserving the most recent 4-week period for data testing.

- BikeNYC: The BikeNYC dataset was generated from February 5, 2014, to August 29, 2014. This dataset contains 6900 traffic flow maps, with dimensions of 16×8 and a temporal resolution of 1 hour. It includes extensive bicycle-related information, including attributes such as trip distance, start and end times, and origin and destination station identifiers. We refer to the last 10 days of the dataset as the test set for evaluation purposes and the previous days as the training set.

5.2. Compared Methods. To measure the overall performance of our approach, we conducted a comprehensive comparative assessment by comparing the MASTGCNet to prominent, well-known models and state-of-the-art methods in the field, including the following:

5.2.1. Conventional Time-Series-Based Models

- HA [19]: Predictions for both passenger flow and traffic congestion can be achieved by using average historical flows at the appropriate time intervals.
- ARIMA [20]: This method involves predicting and understanding various issues related to time series and then tailoring models to address them.

5.2.2. Deep Learning-Based Models

- FC-long short-term memory (LSTM) [21]: It is essentially a fusion of the codec and LSTM models, where each section of the encoder and decoder consists of a certain number of recurrent layers. It is important to note that these recurrent layers contain a defined number of LSTM units. Furthermore, the FC-LSTM model integrates both approaches to predict the result.
- GRU-ED [22]: A model using an encoder–decoder framework based on GRU was used to perform the machine translation task.
- STGCN: This model contains a distinct element called a spatiotemporal block. These blocks are stacked in multiple layers, denoted as k layers, within the model's core. This configuration greatly assists the model in achieving convergence. Ultimately, at both the beginning and end of the process, a 1×1 convolutional layer is employed by the model to aggregate, integrate, and predict various features.
- Graph WaveNet [23]: In this study, the authors offer Graph WaveNet, a unique graph neural network architecture, for spatial–temporal graph-based modeling.
- RPConvformer [24]: The upgraded components of a novel framework named RPConvformer are 1D causal convolutional sequence embedding and relative position encoding.
- ST-ResNet [25]: The ST-ResNet method used a residual-based network to predict urban traffic flows throughout the city. This approach has demonstrated superior performance compared to alternative methods.
- MST3D [26]: This model predicts the vehicle flow throughout the city, the MST3D model employs 3D CNN models to jointly capture multiple spatiotemporal dependencies.
- MASTGCNet: In this work, we combine the GCN and GRU, this framework effectively extracting dynamic features of spatiotemporal from node properties. MASTGCNet implements a resource allocation strategy aimed at minimizing energy consumption during the prediction process.

5.3. Implementation Details and Evaluation Metrics

5.3.1. Preprocessing. In the case of the TaxiBJ dataset, the urban area was partitioned into discrete grid regions measuring 32×32 units, with each temporal segment corresponding to a duration of 30 min. In a parallel manner, for the BikeNYC dataset, we adopted a grid structure comprising dimensions of 16×8 to represent the city map, with each temporal interval set at 1 hour. To make the traffic flows comparable, we scaled them between −1 and 1 using a method called min–max normalization. After the prediction, we adjusted the values to match our research's usual output and compared them with the actual values. The hyperparameter settings are briefly described in Table 5.

5.3.2. Evaluation Metrics. The evaluation of our model performance is conducted through the utilization of two

Table 5: Hyperparameters for MASTGCNet in traffic flow prediction.

Hyperparameter description	Value or range
Keras	2.2.4
TensorFlow	1.13.1
Python	3.6
Learning rate	0.001
Optimization algorithm	Adam
Number of epochs	200
Dropout rate	0.25
Batch size	64
Early stopping patience	15

pivotal metrics: mean average percentage error (MAPE) and root mean square error (RMSE),

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(\widehat{y}_t - y_t\right)^2}, \tag{33}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{\left|\widehat{y}_t - y_t\right|}{y_t}, \tag{34}$$

where the variables y_t and $\widehat{y}_t$ correspond to the actual flow and the predicted flow map, respectively. Additionally, it is important to note that the sample size n is used to validate the resulting prediction results.

5.4. Experiment Results and Analysis. Using the setup and settings mentioned above, we compared how well our MASTGCNet model performed against other competing methods on two real-world datasets: BikeNYC and TaxiBJ. The results are shown in Figures 5 and 6, respectively. The MASTGCNet stands out by achieving the lowest MAPE and RMSE values among all methods, demonstrating its superior performance. Specifically, for BikeNYC, the RMSE is about 4.02, and the MAPE is 18.75%. Similarly, for TaxiBJ, the MAPE is about 14.09 and the RMSE is about 12.43%, respectively. This indicates that our added attention and multiscale feature framework effectively improve the performance of the proposed model. We also use standard deviation, which serves as a valuable metric to assess the consistency and stability of traffic patterns, aiding in anomaly detection, model evaluation, and informed traffic management decisions.

We compared the average results of our method with other methods on two datasets, TaxiBJ and BikeNYC. The results are shown in Table 6. We calculated the average performance for each model by running it 10 times. Table 6 clearly shows that our proposed MASTGCNet model outperforms the other models regarding effectiveness and prediction accuracy. These results highlight the value of our model in understanding the spatiotemporal relationships in predicting traffic flows.

Our MASTGCNet method significantly reduces prediction errors during model training when combined with the spatiotemporal GCN. This demonstrates the adaptability of our method for urban traffic flow prediction. The efficiency of the MASTGCNet model exhibits consistent enhancement, particularly within the domain of long-term traffic flow prediction. In particular, traditional methods, such as ARIMA and HA, struggle to produce highly accurate forecasts, highlighting the limitations of approaches that ignore dynamic spatiotemporal dependencies and focus solely on historical statistical relationships. ARIMA focuses on modeling univariate time series and does not incorporate spatiotemporal correlations between different regions. This limitation prevents the model from exploiting important network structure and road system insights. Incorporating spatial correlations improves regression models such as FC-LSTM and GRU-ED, but they can still not capture the complex dynamic nonlinear spatial and temporal dependencies. The FC-LSTM cannot typically explicitly capture the spatial dependencies in traffic data, which can be crucial for accurate traffic flow prediction in scenarios where road networks play a significant role. Furthermore, our model outperforms the ST-ResNet and MST3D models. The ST-ResNet residual structure is limited in capturing the highly nonlinear relationships in complex traffic dynamics. Similarly, the MS3TD model has a higher number of hyperparameters that require thorough tuning to achieve optimal performance. Insufficient tuning can lead to suboptimal results.

The current methods struggle to predict traffic flow prediction in a given area accurately. Existing approaches such as CNN, RNN, and LSTM manually extract information from images, which reduces prediction accuracy. We need updated methods that automatically gather information using techniques such as the GCN and attention modules. These approaches dynamically gather information from models, improving prediction accuracy while saving time and resources. If we look at the figures (Figures 5 and 6) and compare them with existing methods, it is clear that existing models do not perform as well as our proposed model. Our proposed model achieves better accuracy regarding time, cost, and reliability. It is more reliable and adaptable because it dynamically processes information instead of manually manipulating images. We also combined our model with spatiotemporal techniques to improve training and reduce prediction errors. This combination also speeds up the training time, showing that our model can work well in different situations.

Our model combines the spatiotemporal GCN network to make training more accurate and reduce prediction errors. We tested our model against existing long-term traffic flow prediction methods and found that it performed much better. The existing models used CNN, LSTM, or RNN separately with manual data extraction, which made training longer instead of more efficient. To improve prediction accuracy, we need to extract features from the traffic flow data dynamically. Our model, which uses neural networks such as GCN and RNN with an attention mechanism, allows us to easily estimate traffic flow in a city, including the number of people entering and leaving. This allows us to reduce training time, make predictions more accurate, and save time and cost.

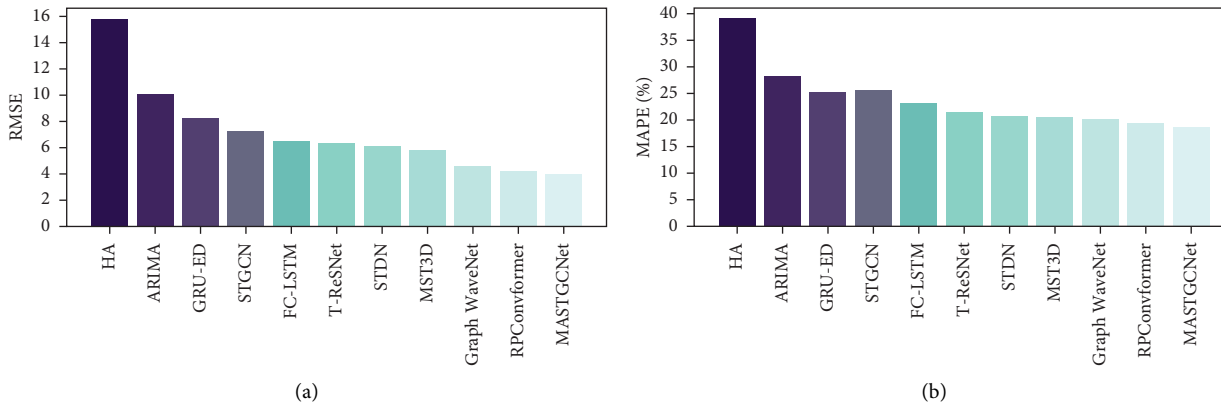

Figure 5: BikeNYC prediction results with MASTGCNet based on (a) RMSE and (b) MAPE (MAPE in %).

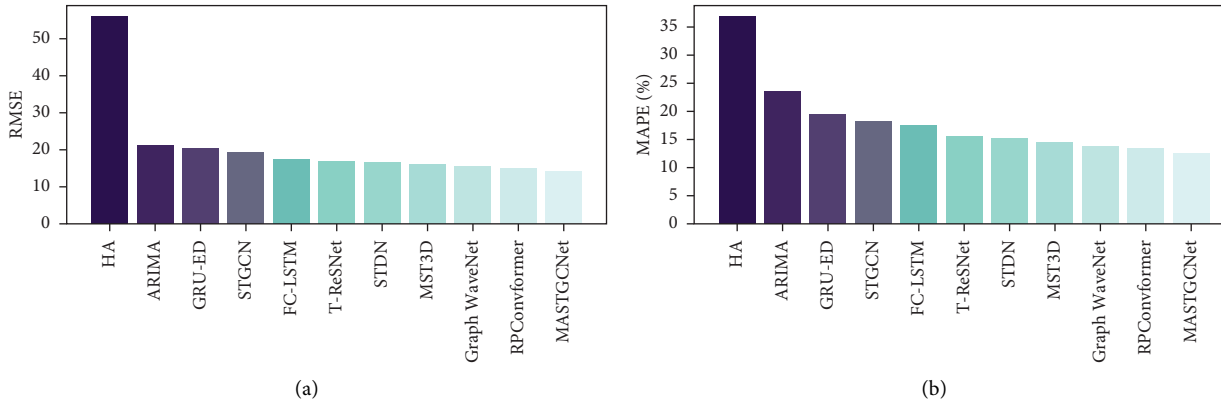

Figure 6: TaxiBJ prediction results with MASTGCNet based on (a) RMSE and (b) MAPE (MAPE in %).

Table 6: Comparisons of models on TaxiBJ and BikeNYC.

Methods	BikeNYC		TaxiBJ	
	RMSE	MAPE (%)	RMSE	MAPE (%)
HA	15.80	38.99	55.99	36.89
ARIMA	10.10	27.99	20.99	23.45
GRU-ED	8.23	25.09	20.18	19.35
STGCN	7.32	25.39	19.18	18.21
FC-LSTM	6.49	22.98	17.32	17.42
ST-ResNet	6.34	21.31	16.89	15.39
STDN	6.20	20.65	16.50	15.15
MST3D	5.81	20.39	15.90	14.41
Graph WaveNet	4.52	20.15	15.26	13.75
RPConvformer	4.27	19.34	14.95	13.24
Our (MASTGCNet)	4.02	18.65	14.09	12.43

In addition, Figure 7 illustrates the prediction performance of various models across different horizons on the TaxiBJ dataset. Our proposed model consistently performs best across all prediction horizons and evaluation metrics, demonstrating its superior ability to capture heterogeneous spatiotemporal dependencies. While STDN shows comparable performance to our model in the first three horizons, its accuracy declines as the prediction horizon increases. This indicates that our model is robust to changes in prediction horizons and underscores the effectiveness of the temporal dependence extraction units in MASTGCNet.

5.5. Results of Different MASTGCNet Variants. To validate the effectiveness of the MASTGCNet method, we conducted a thorough empirical assessment, examining different variants while maintaining consistent experimental setups. Our

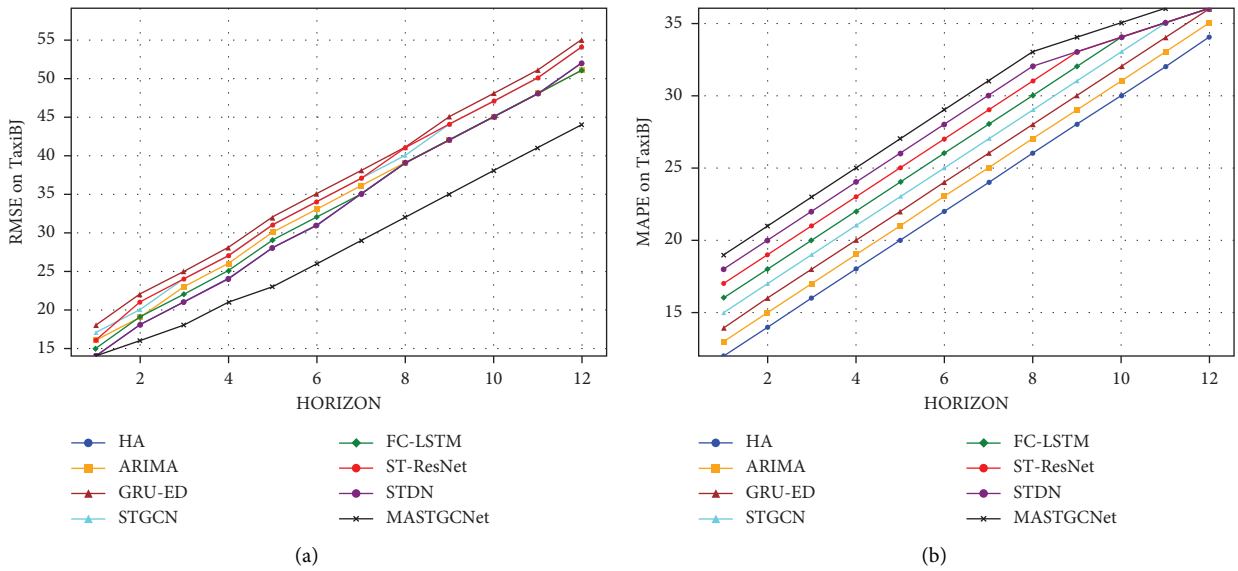

Figure 7: RMSE and MAPE scores at each prediction horizon on the TaxiBJ dataset: (a) RMSE and (b) MAPE.

evaluation focused on three key aspects: (i) assessing the impact of model depth, (ii) analyzing the effects of varying kernel and filter numbers, and (iii) investigating computational complexities, including prediction and training times. The subsequent sections delve into a detailed discussion of these findings, shedding light on our proposed algorithm's performance and computational considerations.

5.5.1. Impact of Depth Model. We conducted an extensive empirical experiment with nine different variations of the MASTGCNet model, each characterized by different depths. We aim to achieve deeper insights into the effectiveness of different depths within the MASTGCNet model. The experimental results are shown in Figure 8. Focusing specifically on the TaxiBJ dataset, Figure 8 provides a detailed visualization of the impact of model depth. Our results show a compelling relationship between the number of residual units representing network depth and the RMSE metric. As we progressively increase the depth of the network by increasing the number of residual units, we observe an initial increase in RMSE, followed by a subsequent increase in RMSE. This trend indicates that deeper networks yield better results. Nevertheless, it is crucial to note that as the number of residual units exceeds a certain threshold, such as ≥ 12, the training process becomes more complex, and the risk of overfitting increases.

5.5.2. Impact of Filter Numbers and Filter Sizes. We have extensively explored different dataset training configurations to optimize filter size selection. Achieving precise prediction accuracy depends on the careful selection of filter size. As shown in Figure 9(b), we observed the profound impact of filter size on the performance of the MASTGCNet methodology. To carefully observe and illustrate the effects of filter size, we systematically adjusted the filter dimensions, ranging from 2×2 to 5×5. Figure 9(b) shows a noteworthy

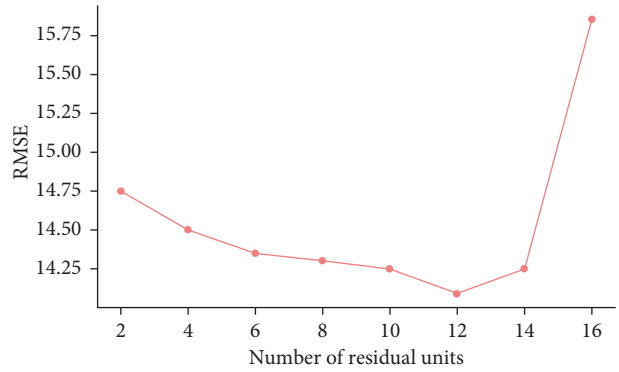

Figure 8: Model depth impact on RMSE through variational analysis. Each model run by 10 times.

trend: an increase in kernel size corresponds to a decrease in the RMSE metric. In addition, as shown in Figure 9(a), the use of larger filter sizes consistently yields significantly better results than the use of smaller filter sizes. This pattern is consistent with our observations regarding filter size and number of filters.

5.6. The Convergence of MASTGCNet Model. Figure 10 illustrates the loss curve of the MASTGCNet model on two real datasets, depicting the training and validation RMSE during the experiment. In Figure 10(a), on the BikeNYC dataset, the RMSE for both the training and validation sets gradually decreases with increased training iterations. Around Iteration 65, the RMSE for both sets stabilizes. In Figure 10(b), for the TaxiBJ dataset, the RMSE for both the training and validation sets also decreases with increasing iterations, reaching stability around Iteration 128.

Figure 11 illustrates that the traffic state matrix varies across different time segments. Initially, the feature matrix before training appears random and unstructured. However,

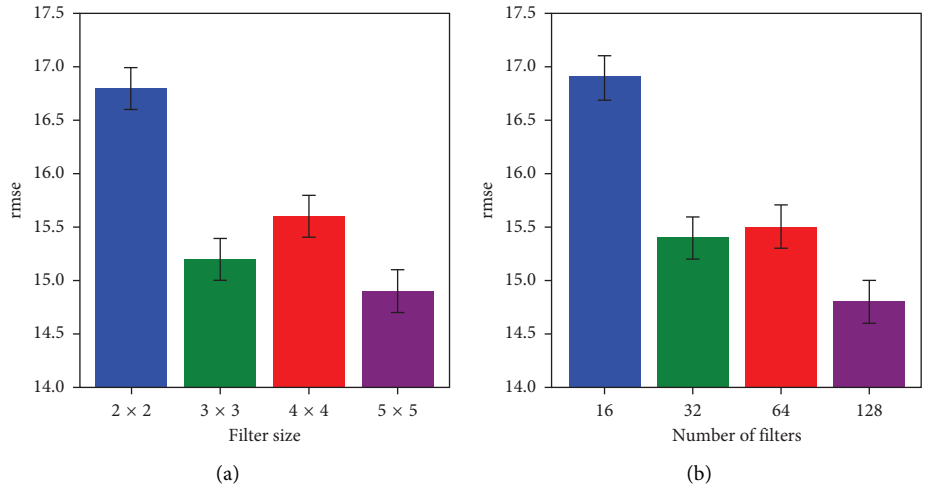

FIGURE 9: Prediction results of various (a) filter sizes and (b) filter numbers.

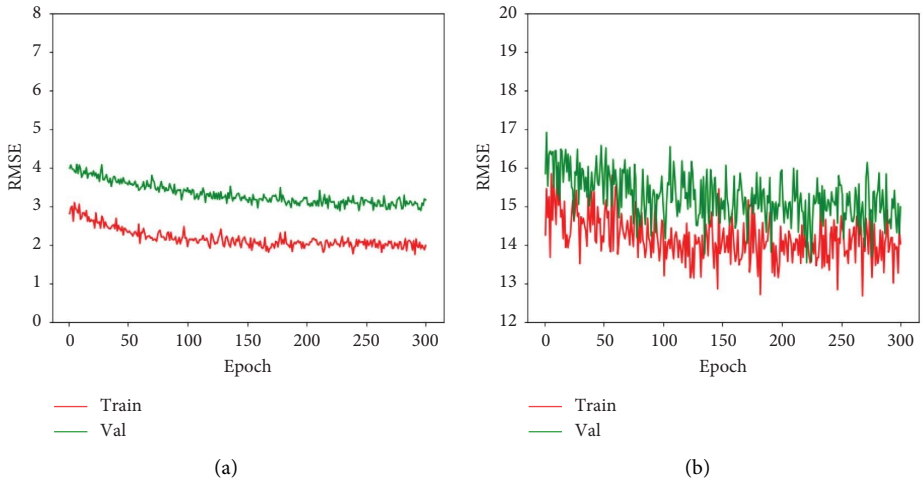

FIGURE 10: Prediction results of both training and validation based on (a) BikeNYC and (b) TaxiBJ datasets.

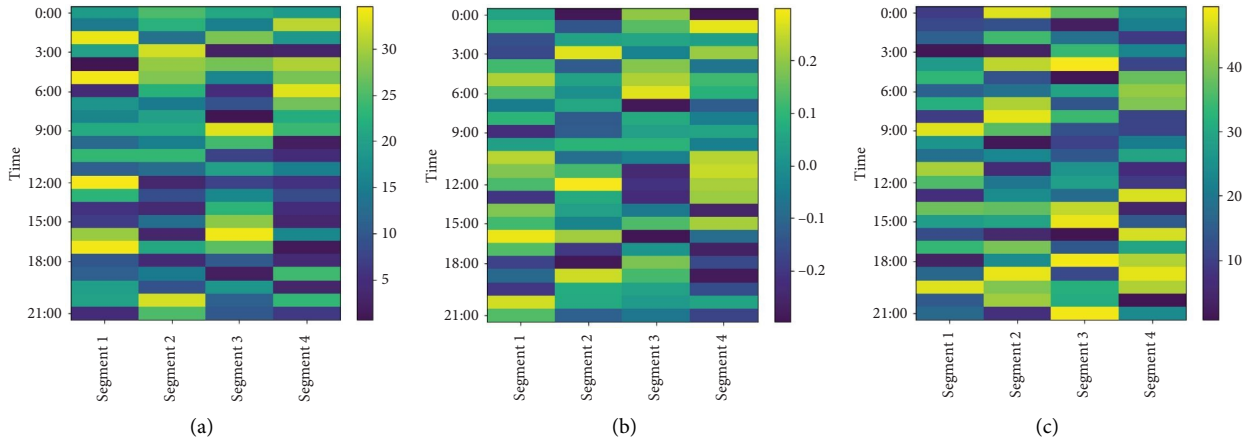

FIGURE 11: Visualization of the traffic state matrix and corresponding feature matrices before and after training. (a) Ground-truth traffic state matrix. (b) Feature matrix before training. (c) Feature matrix after training.

after training, the feature matrix displays distinct temporal patterns similar to the traffic state matrix. This demonstrates that the model successfully learns and abstracts temporal features for prediction.

5.7. Model Efficiency. Subsequently, we conduct a thorough investigation of the performance of our proposed model works both in predicting outcomes and in the training phase, taking into account the response time. We have summarized the results in Table 7. One thing that immediately stands out is that our MASTGCNet model performs better than the STDN and MST3D models in both the training and prediction phases. It is worth noting that the STDN method takes the longest time for both training and prediction. This is because STDN uses certain computations called local CNNs to make predictions, and it also has to go through the entire region using a sliding window. For example, if we work with a dataset like BikeNYC, predicting values for the entire region requires performing a certain computation 16×8 times. On the BikeNYC dataset, our MASTGCNet model significantly outperforms the ST-ResNet approach. This happens mainly because our MASTGCNet method combines two techniques, GCN and GRU, based on GCNs. We see a similar trend on the TaxiBJ dataset. Our MASTGCNet model outperforms both the MST3D and STDN baselines in terms of both prediction speed and training speed. These results demonstrate the advantages of our model.

The time and space complexities of Algorithm 1 are closely related to key parameters, mainly the amount of available historical information and the number of features. In particular, the complexities depend on two crucial factors: the training time, denoted by m, which is the time required to form the dataset D, and the learning time, denoted by n, which characterizes the time required to acquire knowledge within the model while keeping the computational overhead for θ constant. It is important to understand that the algorithm's performance shows different behaviors under different circumstances. In the best-case scenario, the algorithm exhibits a time complexity denoted by $\omega(m+n)$. In this case, the computational cost escalates linearly with both training and learning times. This scenario represents an optimal match of algorithm efficiency with available computational resources, culminating in a favorable performance profile. Conversely, the worst-case scenario reveals a potentially significant time complexity that can rise to $O((mn)(m+n))$ and further simplifies to $O(m^2n + n^2m)$. This particular scenario depends on the number of observations and the dimensionality of the features. This worst-case behavior serves as a stark reminder of the paramount importance of carefully considering resource constraints and dataset characteristics when implementing Algorithm 1.

6. Related Works

In this section, we emphasize how we use deep learning to understand the spatiotemporal correlation for traffic prediction. We also look closer at the concept of attention, which plays an important role in our model.

6.1. Traffic Flow Prediction. In the field of traffic prediction, the incorporation of predictive spatial information is emerging as a key factor. Modern methods using convolution neural networks (CNNs) [27, 28] for traffic flow mapping often rely on grid-based map segmentation. However, the inherent regularity of grid-based data limits their ability to represent complex spatial information effectively. Reference [29] proposed a unique multitask spatiotemporal network for highway traffic flow prediction (MT-STNet) that combines multitask learning, generative inference system, and encoder–decoder structure. References [30, 31] (DCRNN) innovatively presented a diffusion convolution RNN based on distance graphs. Their approach incorporates bidirectional random walks on the graph to capture spatial dependencies.

However, the establishment of spatial topology in these methods through predefined adjacency matrices remains static and limited in nature, with inherent limitations in encapsulating the complex characteristics of complex road networks. Existing methods based on predefined graph structures face significant challenges. First, predefined graph constructs struggle with sparsity issues. Sparsity, which is particularly evident in large graphs, manifests itself in the adjacency matrix, leading to computationally inefficient operations and may fail to encapsulate the totality of spatial dependencies comprehensively. This, in turn, can potentially undermine the accuracy of the model [32]. A secondary concern revolves around the inflexible nature of predefined adjacency matrices, which makes them ill-suited for accommodating dynamic graphs or graphs characterized by recurrent structural changes [33].

In conjunction with the previous discourse on the construction of spatial topology, the strategic modeling of spatial and temporal dependencies emerges as a key factor in traffic prediction. Capturing temporal dependencies requires recourse to rRNNs and their diversification, which are expertly tailored to sequential data with intrinsic capabilities to capture temporal correlations of a temporal nature [34, 35]. The recurrent nature of RNN operations provides increased model flexibility. In Ref. [36], this paper proposes a unique deep learning approach to improve traffic volume forecasting accuracy by addressing these shortcomings. In order to create synthetic data or synthetic traffic volume, a new spatiotemporal dependencies generative adversarial network are first proposed.

In contrast to the more complicated LSTM [37], GRU advocates a simpler and more trainable architecture. Gaining prominence for their effectiveness and versatility, attention mechanisms [38] have found widespread application in various domains. These mechanisms automatically focus on central information extracted from historical input data. Graph attention networks (GANs) [5] have made significant progress in traffic prediction by constructing spatial models. STSGCN [39] introduces a novel concept by devising a synchronized GCN model for spatial and temporal aspects. These modules adeptly capture correlations across space and time. In Ref. [40], to further improve the traffic flow forecasting outcomes, the researchers offer a unique spatial–temporal

Table 7: Training and prediction times for BikeNYC and TaxiBJ.

Methods	BikeNYC		TaxiBJ	
	Training time (s/epoch)	Prediction time (s) (%)	Training time (s/epoch)	Prediction time (s)
STDN	18,973	88.7	369,601	207.4%
MST3D	126	0.23	5902	2.74%
Our (MASTGCNet)	116	0.08	4845	2.57

gated hybrid transformer network (STGHTN), which utilizes global features by transformer and local information from temporal gated convolution and spatial gated graph convolution, respectively. We advised the MASTGCNet model, which combines the GCN with the attention unit to collectively incorporate both the features of spatiotemporal traffic flow, drawing inspiration from previous research as discussed above.

6.2. Attention Mechanism. The attention mechanism has become an important tool in various tasks such as language translation and image recognition, helping to decide which parts are most important to focus on Ref. [41]. This mechanism uses a process where a question and a set of key-value pairs create an output. The output is a mixture of the values, with the importance of each value determined using a compatibility function between the question and its key. Attention mechanisms are generally used in conjunction with convolutional or recurrent networks. In Ref. [42], they present an attention-based spatiotemporal graph attention network (ASTGAT) model for network degradation and oversmoothing issues. In Ref. [43], the model relies on attention to see connections between input and output. Recently, attention mechanisms have also been used in graph neural networks. In Ref. [44], they introduced gated attention networks for graph learning. In Ref. [45], a brand-new GCN dubbed TFM-GCAM incorporates attention mechanisms to better capture nodes' dynamic properties and spatial–temporal attributes.

7. Conclusions and Future Work

In this paper, we proposed a new approach to traffic flow prediction using a deep learning model called MASTGCNet. To address the complexities of traffic, we introduced a dynamic graph generation component and a spatiotemporal multiscale feature extraction component to the standard GCNs. These components help the model learn specific patterns for each point and understand relationships at different levels of detail in the data. We performed experiments using baseline techniques on two extensive traffic flow datasets to confirm the efficiency of our MASTGCNet model. The experimental outcomes affirm the MASTGCNet model's performance and the proposed MUST module. MASTGCNet implements a resource allocation strategy within edge computing to minimize energy consumption during prediction.

In the future, we aim to increase the model simplicity to facilitate its seamless application to related tasks. This approach is intended to strengthen the model's performance and capacity for broader applicability. In addition, we plan to improve our model by integrating external factors, such as unexpected events and weather conditions, to increase its predictive accuracy.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare no conflicts of interest.

Author Contributions

A.A.: conceptualization, software development, and writing – original draft. I.U.: writing – original draft and visualization. S.K.S.: writing – review and editing and revision. A.S.: visualization and revisions. W.J.: visualization, revisions, and writing. H.I.S.: writing – review and revisions. X.B.: conceptualization, software development, writing – original draft, and revisions.

Funding

This work was supported in part by the National Natural Science Foundation of China under Grants 62373255 and 62173234, in part by the Natural Science Foundation of Guangdong Province under Grant 2024A1515011204, in part by the Shenzhen Natural Science Fund through the Stable Support Plan Program under Grant 20220809175803001, and in part by the Open Fund of National Engineering Laboratory for Big Data System Computing Technology under Grant SZU-BDSC-OF2024-15.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62373255 and 62173234, in part by the Natural Science Foundation of

Guangdong Province under Grant 2024A1515011204, in part by the Shenzhen Natural Science Fund through the Stable Support Plan Program under Grant 20220809175803001, and in part by the Open Fund of National Engineering Laboratory for Big Data System Computing Technology under Grant SZU-BDSC-OF2024-15.

References

[1] A. Ali, Y. Zhu, and M. Zakarya, "A Data Aggregation Based Approach to Exploit Dynamic Spatio-Temporal Correlations for Citywide Crowd Flows Prediction in Fog Computing," *Multimedia Tools and Applications* 80, no. 20 (2021): 31401–31433, https://doi.org/10.1007/s11042-020-10486-4.

[2] D. Wang, W. Xu, X. Jia, et al., "Analysis of Intelligent Transportation System Application Based on Internet of Things and Big Data Technology Under the Background of Information Society," *Advances in Multimedia* 2022 (2022): 1–11, https://doi.org/10.1155/2022/6001355.

[3] L. Liu, Y. Tian, C. Chakraborty, et al., "Multilevel Federated Learning-Based Intelligent Traffic Flow Forecasting for Transportation Network Management," *IEEE Transactions on Network and Service Management* 20, no. 2 (2023): 1446–1458, https://doi.org/10.1109/tnsm.2023.3280515.

[4] S. Banerjee, C. Chakraborty, and S. Chatterjee, "A Survey on Iot Based Traffic Control and Prediction Mechanism," *Internet of Things and Big Data Analytics for Smart Generation* (2019), 53–75.

[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," (2017), https://arxiv.org/abs/1710.10903.

[6] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," (2017), https://arxiv.org/abs/1709.04875.

[7] L. Bai, L. Yao, X. Wang, C. Li, and X. Zhang, "Deep Spatial–Temporal Sequence Modeling for Multi-Step Passenger Demand Prediction," *Future Generation Computer Systems* 121 (2021): 25–34, https://doi.org/10.1016/j.future.2021.03.003.

[8] L. Bai, L. Yao, S. S. Kanhere, Z. Yang, J. Chu, and X. Wang, "Passenger Demand Forecasting With Multi-Task Convolutional Recurrent Neural Networks," in *Advances in Knowledge Discovery and Data Mining: 23rd Pacific-Asia Conference, PAKDD 2019* (Macau, China: Springer, April 2019), 29–42.

[9] X. Geng, Y. Li, L. Wang, et al., "Spatiotemporal Multi-Graph Convolution Network for Ride-Hailing Demand Forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence* 33, no. 01 (2019): 3656–3663, https://doi.org/10.1609/aaai.v33i01.33013656.

[10] W. Cao, Y. Wu, C. Chakraborty, D. Li, L. Zhao, and S. K. Ghosh, "Sustainable and Transferable Traffic Sign Recognition for Intelligent Transportation Systems," *IEEE Transactions on Intelligent Transportation Systems* 24, no. 12 (2023): 15 784–815 794, https://doi.org/10.1109/tits.2022.3215572.

[11] A. Ali, Y. Zhu, Q. Chen, J. Yu, and H. Cai, "Leveraging Spatio-Temporal Patterns for Predicting Citywide Traffic Crowd Flows Using Deep Hybrid Neural Networks," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)* (IEEE, 2019), 125–132.

[12] J. D. Hamilton, *Time Series Analysis* (Princeton university press, 2020).

[13] M. J. Cushing and D. I. Rosenbaum, "Historical Averages, Units Roots and Future Net Discount Rates: A Comprehensive Estimator," *Journal of Forensic Economics* 19, no. 2 (2006): 139–159, https://doi.org/10.5085/0898-5510-19.2.139.

[14] M. X. Hoang, Y. Zheng, and A. K. Singh, "Fccf: Forecasting Citywide Crowd Flows Based on Big Data," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2016), 1–10.

[15] B. M. Williams, P. K. Durvasula, and D. E. Brown, "Urban Freeway Traffic Flow Prediction: Application of Seasonal Autoregressive Integrated Moving Average and Exponential Smoothing Models," *Transportation Research Record: Journal of the Transportation Research Board* 1644, no. 1 (1998): 132–141, https://doi.org/10.3141/1644-14.

[16] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," (2016), https://arxiv.org/abs/1609.02907.

[17] X. Bai, M. Cao, W. Yan, and S. S. Ge, "Efficient Routing for Precedence-Constrained Package Delivery for Heterogeneous Vehicles," *IEEE Transactions on Automation Science and Engineering* 17, no. 1 (2020): 248–260, https://doi.org/10.1109/tase.2019.2914113.

[18] A. Ali, I. Ullah, M. Shabaz, et al., "A Resource-Aware Multi-Graph Neural Network for Urban Traffic Flow Prediction in Multi-Access Edge Computing Systems," *IEEE Transactions on Consumer Electronics* 70, no. 4 (2024): 7252–7265, https://doi.org/10.1109/tce.2024.3439719.

[19] E. Zivot and J. Wang, "Vector Autoregressive Models for Multivariate Time Series," *Modeling Financial Time Series With S-PLUS®* (2006), 385–429.

[20] B. M. Williams and L. A. Hoel, "Modeling and Forecasting Vehicular Traffic Flow as a Seasonal Arima Process: Theoretical Basis and Empirical Results," *Journal of Transportation Engineering* 129, no. 6 (2003): 664–672, https://doi.org/10.1061/(asce)0733-947x(2003)129:6(664).

[21] Y. Tong, Y. Chen, Z. Zhou, et al., "The Simpler the Better: A Unified Approach to Predicting Original Taxi Demands Based on Large-Scale Online Platforms," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2017), 1653–1662.

[22] T. Chen and C. Guestrin, "Xgboost: A Scalable Tree Boosting System," in *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining* (ACM, 2016), 785–794.

[23] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph Wavenet for Deep Spatial-Temporal Graph Modeling," *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (2019), 1907–1913, https://doi.org/10.24963/ijcai.2019/264.

[24] Y. Wen, P. Xu, Z. Li, W. Xu, and X. Wang, "Rpconvformer: A Novel Transformer-Based Deep Neural Networks for Traffic Flow Prediction," *Expert Systems With Applications* 218 (2023): 119587, https://doi.org/10.1016/j.eswa.2023.119587.

[25] J. Zhang, Y. Zheng, and D. Qi, "Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction," in *AAAI* (2017), 1655–1661.

[26] C. Chen, K. Li, S. G. Teo, et al., "Exploiting Spatio-Temporal Correlations with Multiple 3d Convolutional Neural Networks for Citywide Vehicle Flow Prediction," in *2018 IEEE International Conference on Data Mining (ICDM)* (IEEE, 2018), 893–898.

[27] J. Zhu, C. Tao, H. Deng, et al., "Ast-gcn: Attribute-Augmented Spatiotemporal Graph Convolutional Network for Traffic Forecasting," (2020), https://arxiv.org/abs/2011.11004.

[28] E. H. Bouzidi, A. Outtagarts, R. Langar, and R. Boutaba, "Deep Q-Network and Traffic Prediction Based Routing Optimization in Software Defined Networks," *Journal of*

Network and Computer Applications 192 (2021): 103181, https://doi.org/10.1016/j.jnca.2021.103181.

[29] G. Zou, Z. Lai, T. Wang, Z. Liu, and Y. Li, "Mt-stnet: A Novel Multi-Task Spatiotemporal Network for Highway Traffic Flow Prediction," *IEEE Transactions on Intelligent Transportation Systems* 25, no. 7 (2024): 8221–8236, https://doi.org/10.1109/tits.2024.3411638.

[30] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," (2017), https://arxiv.org/abs/1707.01926.

[31] X. Bai, Y. Ye, B. Zhang, and S. S. Ge, "Efficient Package Delivery Task Assignment for Truck and High Capacity Drone," *IEEE Transactions on Intelligent Transportation Systems* 24, no. 11 (2023): 13422–13435, https://doi.org/10.1109/tits.2023.3287163.

[32] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting," *Advances in Neural Information Processing Systems* 33 (2020): 17 804–817 815.

[33] F. Li, J. Feng, H. Yan, et al., "Dynamic Graph Convolutional Recurrent Network for Traffic Prediction: Benchmark and Solution," *ACM Transactions on Knowledge Discovery from Data* 17, no. 1 (2023): 1–21, https://doi.org/10.1145/3532611.

[34] Y. Wu, Y. Xiang, T. Baker, et al., "Collaborative Attack Sequence Generation Model Based on Multiagent Reinforcement Learning for Intelligent Traffic Signal System," *International Journal of Intelligent Systems* 2024, no. 1 (2024): 4734030, https://doi.org/10.1155/2024/4734030.

[35] X. Zhang, C. Huang, Y. Xu, et al., "Traffic Flow Forecasting With Spatial-Temporal Graph Diffusion Network," *Proceedings of the AAAI Conference on Artificial Intelligence* 35, no. 17 (2021): 15 008–015 015, https://doi.org/10.1609/aaai.v35i17.17761.

[36] K. Zhu, S. Zhang, J. Li, D. Zhou, H. Dai, and Z. Hu, "Spatiotemporal Multi-Graph Convolutional Networks With Synthetic Data for Traffic Volume Forecasting," *Expert Systems With Applications* 187 (2022): 115992, https://doi.org/10.1016/j.eswa.2021.115992.

[37] Q. Wang, J. Yang, and H. Song, "A Novel Long Short-Term Memory Learning Strategy for Object Tracking," *International Journal of Intelligent Systems* 2024, no. 1 (2024): 6632242, https://doi.org/10.1155/2024/6632242.

[38] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "Geoman: Multi-Level Attention Networks for Geo-Sensory Time Series Prediction," *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* 2018 (2018): 3428–3434, https://doi.org/10.24963/ijcai.2018/476.

[39] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence* 34, no. 01 (2020): 914–921, https://doi.org/10.1609/aaai.v34i01.5438.

[40] J. Liu, Y. Kang, H. Li, H. Wang, and X. Yang, "Stghtn: Spatial-Temporal Gated Hybrid Transformer Network for Traffic Flow Forecasting," *Applied Intelligence* 53, no. 10 (2023): 12 472–512 488, https://doi.org/10.1007/s10489-022-04122-x.

[41] X. Yan, X. Gan, R. Wang, and T. Qin, "Self-Attention Eidetic 3d-Lstm: Video Prediction Models for Traffic Flow Forecasting," *Neurocomputing* 509 (2022): 167–176, https://doi.org/10.1016/j.neucom.2022.08.060.

[42] Y. Wang, C. Jing, S. Xu, and T. Guo, "Attention Based Spatiotemporal Graph Attention Networks for Traffic Flow Forecasting," *Information Sciences* 607 (2022): 869–883, https://doi.org/10.1016/j.ins.2022.05.127.

[43] K. Kim, S. Jin, S. Ko, and J. Choo, "Stgrat: A Spatio-Temporal Graph Attention Network for Traffic Forecasting," *CIKM* (2020).

[44] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "Gaan: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs," (2018), https://arxiv.org/abs/1803.07294.

[45] J. Chen, L. Zheng, Y. Hu, W. Wang, H. Zhang, and X. Hu, "Traffic Flow Matrix-Based Graph Neural Network with Attention Mechanism for Traffic Flow Prediction," *Information Fusion* 104 (2024): 102146, https://doi.org/10.1016/j.inffus.2023.102146.