Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Online transfer learning with an MLP-assisted graph convolutional network for traffic flow forecasting: a solution for edge intelligent devices*

Jingru SUN[†‡1], Chendingying LU[1], Yichuang SUN[2], Hongbo JIANG[1], Zhu XIAO[1]

[1]*College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China*

[2]*School of Engineering and Computer Science, University of Hertfordshire, Hatfield AL10 9AB, United Kingdom*

[†]E-mail: jt_sunjr@hnu.edu.cn

Received Dec. 12, 2024; Revision accepted Mar. 13, 2025; Crosschecked

Unedited

Abstract: Traffic flow prediction is crucial for intelligent transportation and aids in route planning and navigation. However, existing studies often focus on prediction accuracy improvement, while neglecting external influences and practical issues like resource constraints and data sparsity on edge devices. We propose an online transfer learning (OTL) with multi-layer perceptron (MLP)-assisted graph convolutional network (GCN) framework (OTL-GM), consisting of two parts: transferring source domain features to edge devices and using online learning to bridge domain gaps. Experiments on four datasets demonstrate OTL's effectiveness, and in a comparison with a model without OTL, convergence time of the OTL-model increased from 24.77% to 95.32%.

Key words: Online transfer learning; Traffic forecasting; Intelligent edge devices
https://doi.org/10.1631/FITEE.2401059 **CLC number:**

1 Introduction

The integration of the internet of things (IoT) in intelligent transportation systems (ITSs) signifies a transformative era with unprecedented possibilities (Bojan et al., 2014). A widespread sensor network in urban traffic nodes enables real-time monitoring of variables like traffic flow and congestion, and swiftly transmits data for dynamic traffic management (Zhang and Lu, 2020). The significance of IoT lies in processing data with advanced techniques, such as deep learning, and providing a scientific foundation for optimized transportation systems with heightened efficiency and reduced congestion. Accurate traffic flow prediction is crucial for intelligent transportation, empowering advanced management, and optimizing resources for efficiency. However, predicting delay is a more important aspect for traffic managers and traveling vehicles (Hashemi and Abdelghany, 2015). Excessively long delays can lead to wrong decisions by traffic managers and mislead vehicles in planning their driving routes. The synergy of traffic flow prediction and IoT establishes an intelligent foundation for resilient, efficient, and secure urban transportation systems (Derawi et al., 2020). However, some contradictions have arisen during the integration of IoT and ITS.

On the one hand, IoT devices have evolved from simple magnetic and infrared sensors to the current generation of intelligent edge devices (Qadri et al., 2020). Initially, basic motion detection relied on magnetic and infrared technologies. Later, cameras were introduced to improve monitoring preci-

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 62171182), the Natural Science Foundation Project of Chongqing, Chongqing Science and Technology Commission (No. CSTB2022NSCQ-MSX5843), the Science and Technology Progress and Innovation Project of Hunan Transportation Department of China (No. 202306)

ORCID: Jingru SUN, https://orcid.org/0000-0001-9474-7778

© Zhejiang University Press 2025

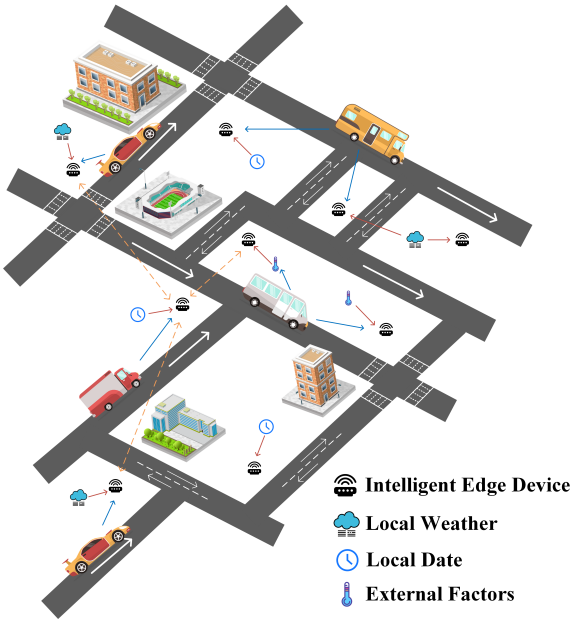


Fig. 1 The real-world scenario of traffic collection and prediction.

sion. However, integration among different sensors is still limited. Intelligent edge devices now have integrated sensors, computing abilities, and connectivity to provide advanced real-time processing and feedback functions. This integration is driving the IoT towards greater intelligence and efficiency. However, such multi-functional devices are often limited by resources, especially computing resources (Asim et al., 2020). They not only compute, but transmit wireless data (Cong et al., 2021) and information on road condition detection (Singh et al., 2021). Therefore, how to reduce computing resource consumption and provide as many computing resources as possible for other aspects has become an urgent problem to be solved. In addition, collecting traffic flow data can be challenging in extreme weather conditions (Datla and Sharma, 2008).

On the other hand, as the core of traffic flow prediction problems, prediction accuracy has always been a hot topic in academic research. In the 1990s, traffic flow prediction based on micro-simulation and macro-models began to attract attention (Bando et al., 1995). During the 2010s, there was increased recognition of the significance of data-driven approaches and real-time technology (Nellore and Hancke, 2016). Currently, deep learning has emerged as an essential technology for predicting traffic flow, highlighting the superiority of neural network models in handling complex spatiotemporal relationships (Kashyap et al., 2022). Various models have emerged for spatiotemporal prediction, such as the spatial-temporal graph convolutional Network (STGCN) (Yu et al.), diffusion convolutional recurrent neural network (DCRNN) (Li et al., 2018), and a graph attention method (Tang et al., 2020) , all of which achieve excellent accuracy. The pursuit of higher accuracy is accompanied by increased algorithmic complexity and prediction time (Zhu et al., 2021; Diao et al., 2019; Guo et al., 2022).

The increasing complexity of algorithms conflicts the limited computing resources of IoT intelligent edge devices. In the context of traffic flow prediction, this issue becomes even more apparent.

As shown in Fig. 1, numerous intelligent transportation facilities have recently been deployed on urban roads, such as camera modules, geomagnetic modules, temperature sensors, and 5G modules for intercommunication (as shown in Fig. 2). They can accurately ascertain the traffic flow, detect temperature, and other information. From the real-world perspective, traffic flow is affected by multiple factors, such as different types of weather, which lead to differences in data distribution. When facing extreme weather conditions such as blizzards and heavy fog, people living in cities surely tend to stay in buildings rather than go out. Such extreme weathers reduces the accuracy of detection algorithms like YOLO, which leads to data sparsity. Similarly, different periods exhibit changes in data distribution (proven in Fig. 3). Morning and evening rush hours can lead to a sharp increase in traffic flow, which is a nonlinear characteristic. Also, weekends and holidays can affect people's willingness to travel. Therefore, handling nonlinear characteristics and alleviating the limited resources of intelligent edge devices has become a challenge.

Traditional traffic flow prediction algorithms often consider all features in the dataset (Zhang and Patras), which introduces many irrelevant data features and leads to reduced prediction accuracy. Only a few studies (Sun et al., 2022) address issues related to regional structuring. Based on the above analysis, there remain two challenges in traffic flow prediction: (1) limitations on computational resources and (2) the sparsity of data caused by extreme weather and the accuracy of object detection algorithms. Therefore, existing literature is insufficient to address these emerging challenges and problems in the real-world

Unedited

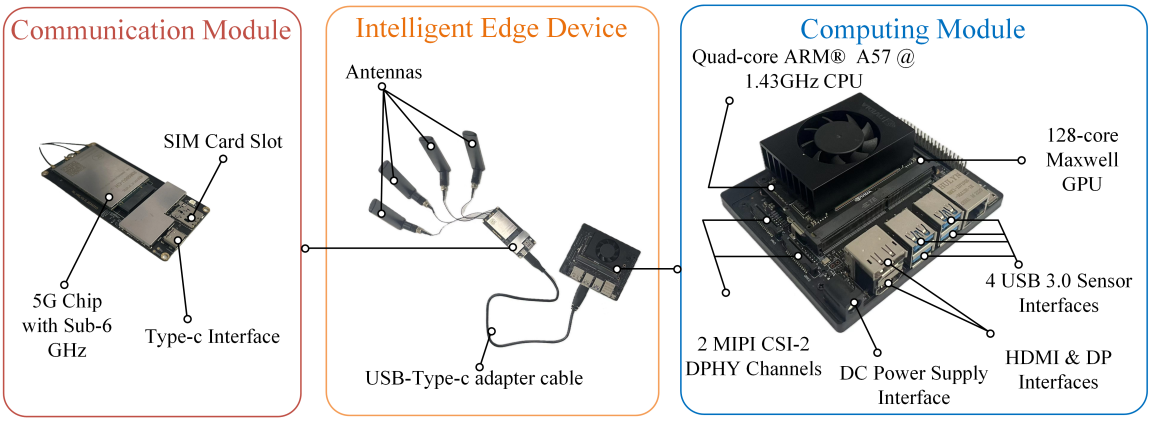


Fig. 2 Intelligent edge device with a 5G communication module and various interfaces

scenario. To solve these problems, we proposed an online transfer learning (OTL) framework with an multi-layer perceptron (MLP)-assisted graph convolutional network (GCN) (OTL-GM). The main contributions of this paper can be summarized as follows:

- To ensure the the models running on intelligent edge devices are lightweight, we combined the temporal model MLP with a spatial model GCN to obtain a GM model that captures spatio-temporal features.
- We combined the GM model with a novel OTL framework for traffic forecasting, which not only relieves the stress on computational resources but also maintains prediction accuracy at current research levels due to opting for proper models with smaller Euclidean distance.
- We validated the effectiveness of our OTL framework with other methods by conducting extensive experiments on the real-world dataset PeMS04, which is classified into three datasets of different data types. Algorithm convergence time has been greatly improved. To the best of our knowledge, it is the first attempt to apply OTL for traffic flow forecasting.

The remainder of the article is organized as follows: Section 2 provides some related work about OTL and traffic flow forecasting with spatio-temporal correlation. Before introducing the model, Section 3 provides the prerequisite knowledge and defines traffic flow prediction in detail. Section 4 mainly introduces the details of the model and algorithm. Section 5 presents the validation of the effectiveness of the proposed OTL-GM and discusses the experimental results. Finally, Section 6 summarizes the paper.

Unedited

2 Related work

2.1 Traffic flow forecasting

There are two main categories of methods for predicting traffic flow: model-driven approaches and data-driven approaches. Model-driven approaches use statistical theory to create a comprehensive model that learns the time feature and analyzes the data. There are several typical model-driven approaches, such as the history average (HA) model (Liu and Guan), the auto-regressive moving average model (Williams and Hoel, 2003), and the vector auto-regressive model (Zivot and Wang, 2006). In particular, (Hamed et al., 1995) considered the flow of each road to obtain a proper ARIMA model. However, model-driven approaches are insufficient for dealing with real-world traffic data because traffic flow is impacted by various factors and is nonlinear.

To tackle these issues, data-driven approaches have gained traction in recent decades. Data-driven approaches include machine learning and deep learning. Machine learning approaches, in time order, are k-nearest neighbor (Van Lint and Van Hinsbergen, 2012) and support vector regression (Jeong et al., 2013). However, the fatal drawback of machine learning is that it is excessively dependent on setting initial hyperparameters, that is, prior knowledge, which determines the accuracy of the predic-

tion. The complexity of traffic data in the real network is also vital. Hence, the current research has shifted toward deep learning and indicates that it can improve prediction accuracy related to the high complexity of traffic data and represent traffic features without prior knowledge, such as the deep belief network (DBN) (Jia et al., 2016), stacked auto-encoder (Lv et al., 2014) and MLP (Rumelhart et al., 1986).

2.2 Spatio-temporal correlation

Given this information, traffic flow forecasting is a time series prediction problem indeed. Some neural networks for time series forecasting problems are applied to traffic flow prediction, such as the recurrent neural network (RNN) (Gräber et al., 2019) and the time complexity is $O(T)$, where T represents the length of the input sequence. Furthermore, to overcome vanishing gradient problem and the exploding gradient of the RNN, derivatives of the RNN, such as the long short-term memory (LSTM) (Zhang and Guo, 2022) neural network. Actually, the time complexity of LSTM is $O(4T)$ because of its three gates, but according to the rules of algorithm complexity, it is also still $O(T)$. Gated recurrent unit (GRU) networks (Sutskever et al.) have also been introduced with $O(T)$ time complexity. Hence, the temporal feature can be represented properly. However, the performance of LSTM in predicting long sequences falls short of expectations, which prompted the introduction of the transformer (Vaswani et al.) at the cost of the time complexity $O(T^2)$. Its parallelized self-attention mechanism enables efficient processing. However, the increased computational demands can not meet the requirement for limited computational resources. We used a MLP and adjusted its width and depth to reduce time complexity while ensuring a certain level of accuracy. The time complexity of GM is $O(\sum_{l=1}^{L} I_l O_l)$, where L is the number of layers, I_l is the input dimension of layer l and O_l is the output dimension of layer l. The final time complexity is a constant level $O(1)$.

Actually, the traffic flow is influenced by not only the time dimension but also the spatial dimension, which is known as spatio-temporal correlation. To take the spatial feature into consideration, convolutional neural networks (CNNs) (Krizhevsky et al., 2017) have been applied to extract the local feature. For example, MGSTC (Chen et al., 2020) established multiple gated spatio-temporal CNN networks to extract multiple spatio-temporal dependencies and S-TCN (Guo et al., 2023) is modeled with spatio-temporal CNN to capture several correlations among different dimensions. However, CNN can only be applied to data structures with Euclidean properties, that is regular grid shapes, such as images, which is obviously not suitable for capturing spatial features in transportation networks due to the complex network topology. Therefore, we adopt a GCN to capture the spatial features, which means dealing with a non-Euclidean topology graph and reducing the parameters based on the same adjacency matrix. Excellent work in recent years includes the primary GCN (Kipf and Welling, 2017), which found that 2-3 layers obtained the best results; spatial-temporal GCN (Yu et al.), which combined gated CNNs and a GCN; and the temporal GCN (Zhao et al., 2020), which combined GRUs and a GCN. By contrast, we use a two-layer GCN and a three-layer MLP to ensure a lightweight model and capture the spatio-temporal features.

2.3 Transfer learning

The motivation for adopting transfer learning in some literature (Mallick et al., 2021) is to solve the problem of data sparsity due to the poor urban infrastructure. However, even with the development of cities with more intelligent roadside equipment, there will still be data sparsity problems due to the existence of extreme weather such as heavy fog and blizzards (Datla and Sharma, 2008). We take these weather factors into account in our transfer learning. For effective transfer of knowledge from one trained model to another, we need to select a relatively similar model that has a similar data distribution. Nonetheless, transfer learning leads to prediction error to some extent. Several studies have used different methods to solve this problem. (Yao et al., 2023) proposed an adversarial domain adaptation to reduce the error caused by transfer learning. (Kwon et al.) use incremental learning to help deep learning models learn new tasks continuously without forgetting what was learned previously. (Tzeng et al., 2015) adopted an approach to simultaneously optimize for domain invariance to facilitate domain transfer and used a soft label distribution matching loss to transfer information between tasks. We use Euclidean distance to characterize the similarity between data distributions and implement on-

Unedited

line learning to minimize transfer learning errors by using intelligent roadside equipment to gather real-time data.

2.4 Online learning

Online learning has been investigated considerably during the past decades to achieve improved prediction accuracy. In this study, we use online learning to eliminate the errors caused by transfer learning. Much research has focused on the criterion of maximum-margin (Li and Long; Gentile, 2000; Crammer et al., 2006). Their studies are all devoted to minimizing the algorithm loss. One of the most classic algorithms (Rosenblatt, 1958) constantly adjusted the weight of the classifier based on the latest samples. Later, a new algorithm named double updating online learning (DUOL) emerged (Zhao et al., 2011), which is designed to update not only the weights of current instances but also the weights of instances that have already been received. Recently, (Han et al., 2020) adopted the approach of online learning to minimize the model training time. However, these methods are inconsistent with our purpose of reducing the error caused by transfer learning. Consequently, we use real-time data collection to update the parameters of the last fully connected layer, so online learning can reduce prediction errors and training time, thus decreasing resource consumption.

3 Preliminary information

This section introduces some important notations (Table 1), definitions that are used in the model, and defines the task of the prediction model.

3.1 Traffic flow data

The traffic flow data $X_t \in \mathbb{R}^{N\times C}$ are collected by node (as shown in Fig. 1) at time step t, where C represents the kinds of traffic data (generally speaking, it may be traffic volume, average speed or road occupancy) and N denotes the whole node. Considering a batch of data of length τ as input, we can obtain $\mathcal{X} = (X_{t_1}, X_{t_2}, ..., X_{t_\tau}) \in \mathbb{R}^{\tau\times N\times C}$. After feeding the input into the network, we get the predicted results $\mathcal{Y} = (Y_{t_\tau+1}, Y_{t_\tau+2}, ..., Y_{t_\tau+t_\lambda}) \in \mathbb{R}^{\lambda\times N\times C}$.

Table 1 Key notations

Symbols	Description
G	The graph of the road network.
X	Data collected by the devices.
$\mathcal{X}$	A batch of collected data.
$\mathcal{Y}$	Predicting results.
τ	Length of $\mathcal{X}$.
λ	Predicting step size.
C	Numbers of kinds of data.
N	Numbers of the node in the network.
V	The set of nodes.
E	The set of edges between any two nodes.
A	Adjacent matrix of G .
T	Time dimension.
S	Spatial dimension.
$\hat{A}$	Graph matrix with added self-connection
I_N	The identity matrix
$\widetilde{D}$	Degree matrix with added self-connection
D	Degree matrix
$H^{(l)}$	The output of lth layer
$\theta^{(l)}$	The weight parameter
σ	Sigmoid activation function

3.2 Road network structure

In this study, the structure of the traffic network can be regarded as a weighted directed graph $G = (V, E, A)$, where V represents all the nodes in the road network. Generally, vertex, node, sensor, and detector are all viewed as the same thing. For simplicity, we use node for the following description, which is denoted as $v_i \in V, i \in (1, 2, ..., N)$; E is the set of the edges in the whole network, which indicates the edges between any two nodes; $A \in \mathbb{R}^{N\times N}$ denotes the adjacency matrix of graph G.

3.3 Definition of traffic forecasting

Because traffic flow prediction is essentially a time series prediction problem, given the input $\mathcal{X} = (X_{t_1}, X_{t_2}, ..., X_{t_\tau}) \in \mathbb{R}^{\tau\times N\times C}$, we aim to obtain the result $\mathcal{Y} = (Y_{t_\tau+1}, Y_{t_\tau+2}, ..., Y_{t_\tau+t_\lambda}) \in \mathbb{R}^{\lambda\times N\times C}$, where t_λ represents the predicting horizons. The problem is defined as follows:

$$\mathcal{Y} = \underset{Y_{groundTruth}}{\arg\max} \ \log P(Y_{groundTruth}|\mathcal{X}) , \quad (1)$$

where $Y_{groundTruth}$ represents the ground truth observed by intelligent devices and $P(\cdot|\cdot)$ is conditional probability function.

Unedited

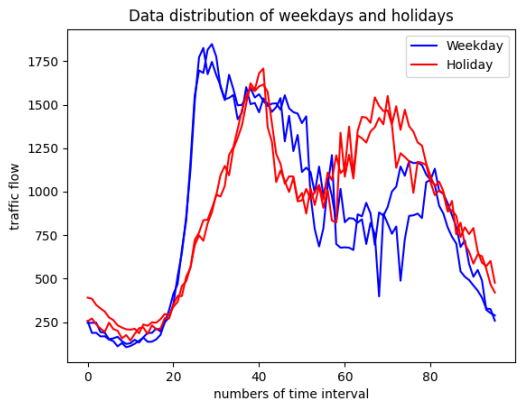


Fig. 3 This dataset collects real-time traffic data every 15 min, which means there are 96 collection points 24 h a day corresponding to the numbers on the time interval-axis. The traffic-flow axis indicates how many cars there are in a 15-min period

3.4 Transfer learning

To demonstrate the universality of the similarity in data distribution across various datasets, we extract similar data distributions in the data set (located near Heathrow Airport on the M25 highway in the UK, URL: `http://tris.highwaysengland.co.uk/detail/trafficflowdata`) and visualize some of the same type of data. Taking weekday data and holiday data as examples, from a qualitative perspective, we find that the same type of data has similar distribution in amplitude and phase (as shown in Fig. 3), whereas different types of data have different phase and amplitude. From a quantitative perspective, we use Euclidean distance to determine the similarity of samples

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}, \tag{2}$$

where x_i and y_i represent different sample points at the same time and n is the total number of samples. In this case, after calculation, we have the Table 2, in which the decimal part of the data has been omitted.

Table 2 Euclidean Distance Between Each Other

	Holi1	Holi2	Week1	Week2
Holi1	0	1602	3896	3582
Holi2	1602	0	4086	3739
Week1	3896	4086	0	1366
Week2	3582	3739	1366	0

Obviously, the Euclidean distance of the same type data is much smaller than that of the different type of data. Based on the results above and the real-time interaction of the intelligent edge devices, we choose transfer learning to obtain the pretrained models and choose the most suitable one. First, we trained multiple models separately depending on the data type using edge servers or other servers with sufficient computing resources. A type of model corresponds to a type of data distribution. Then we obatined the signal (type of data and weather) from intelligent devices, and calculated the Euclidean distance between the data distribution of pretrained models and the data type represented by the signal. Finally, we chose the smallest one to transfer.

3.5 Online transfer learning

The transfer learning discussed in the previous section only involves transferring a model with the most suitable data distribution. However, the distribution of real-time data may not align closely with the transferred model, so it is necessary for smart edge devices to continuously collect real-time data and update some parameters of the model's layers to enhance prediction accuracy and timeliness.

4 Proposed model and algorithm

This section introduces the details of the model and algorithm steps.

4.1 Overview of the GM framework

In this section, we explain the proposed GM architecture. As shown in Fig. 4, GM is composed of spatial and temporal models, which are used to extract spatial and temporal features respectively. The whole input $\mathcal{X}=(X_{t_1},X_{t_2},...,X_{t_\tau})\in\mathbb{R}^{\tau\times N\times C}$ is divided into small batches $\mathcal{X}_\mathcal{B}=(X_{t_1},X_{t_2},...,X_{t_B})\in\mathbb{R}^{B\times t_B\times N\times C}$, and B and t_B represent the numbers of batches and the size of each batch. The W_{Best-S} and W_{Best-T} are the parameters of the best model transferred by our online transfer learning framework. α and β represent the weight of time features and space features respectively. The final prediction result $Y=\alpha Y^{Spatial}+\beta Y^{Temporal}$. The detail of each module and the online transfer learning algorithm are described as follows.

Unedited

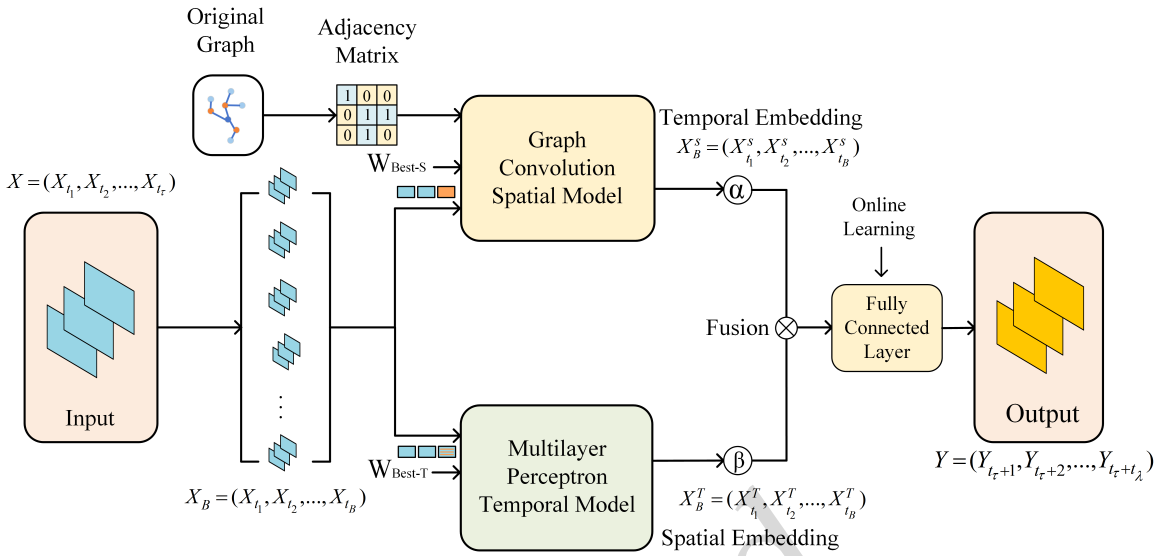


Fig. 4 Entire architecture of our proposed GM

Unedited

4.2 GCN for the Spatial Feature

According to the previous discussions, spatial feature extraction is crucial for traffic flow prediction. CNNs aim at data with Euclidean properties, such as images and other regular grids. However, the topological structure of both urban and highway structures have non-Euclidean properties, which means CNNs no longer work in such cases. Based on this problem, some literature (Kipf and Welling, 2017) gradually evolved CNNs into GCNs and gained much attention. GCNs were originally used in many fields such as document classification (Defferrard et al.) and image classification (Bruna et al., 2014). Observing the formula of graph convolution in (Kipf and Welling, 2017), we find that as long as there is an adjacency matrix and a feature matrix, a filter can be generated in the Fourier domain that can extract spatial features between nodes by their first-order neighborhood. Guided by the prediction result, we can stack such filters to establish the GCN model, which is described as follows:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\hat{A}\widetilde{D}^{\frac{1}{2}}H^{(l)}\theta^{(l)}), \tag{3}$$

where $\hat{A} = A + I_N$ is the graph matrix with added self-connection, A is the adjacency matrix of the graph and I_N is the identity matrix. $\widetilde{D} = D + I_N$ is D with added self-connection, $D = \sum_j \hat{A}_{i,j}$ is the degree matrix of the graph, $H^{(l)}$ is the output of the lth layer, $\theta^{(l)}$ represents the weight parameter

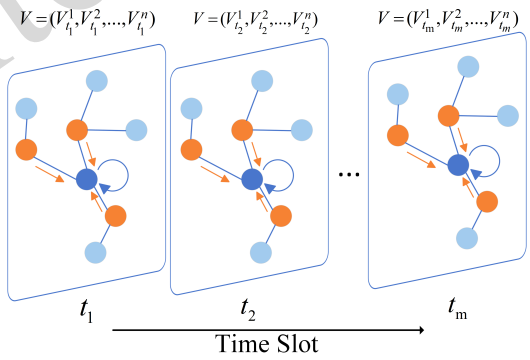


Fig. 5 Process of graph convolution aggregation

corresponding to that layer, and $\sigma(\cdot)$ is the sigmoid function for a nonlinear activation:

$$\sigma(x) = \frac{e^x}{e^x + 1}, \tag{4}$$

As shown in Fig. 5, the notion above the figure $V = (V_{t_i}^1, V_{t_i}^2, ..., V_{t_i}^n), i \in (1, 2, ..., m)$ represents the traffic value of each node in the graph at time t_i. Each node aggregates its first-order neighborhood and itself every time interval, and obtains a result that only correlates with spatial features in the next time interval. Take the deep blue dot as an example, the orange node as its first-order neighborhood, affects its result at the next time slot. The deep blue node also affects itself. When only considering points in the first-order neighborhood, the light blue points, as the second-order neighborhood are not considered.

For the input data $\mathcal{X} = (X_{t_1}, X_{t_2}, ..., X_{t_\tau}) \in \mathbb{R}^{\tau \times N \times C}$, we construct an adjacency matrix $A \in \mathbb{R}^{N \times N}$ and degree matrix $D \in \mathbb{R}^{N \times N}$ based on the node relationships given by the PeMS04. According to Eq. (3) we get the output of the hidden layer:

$$H^{(l)} = \sigma(\widetilde{D}^{-\frac{1}{2}} \hat{A} \widetilde{D}^{\frac{1}{2}} H^{(l-1)} \theta^{(l)}) \in \mathbb{R}^{\tau \times N \times C}, \quad (5)$$

and the output of the GCN:

$$Y^{Spatial} = \sigma(\widetilde{D}^{-\frac{1}{2}} \hat{A} \widetilde{D}^{\frac{1}{2}} H^{(last-1)} \theta^{(last)}) \in \mathbb{R}^{\tau \times N \times C}. \quad (6)$$

4.3 MLP for temporal feature

As mentioned previously, there are many excellent models for capturing temporal features, such as RNN, LSTM (Zhang and Guo, 2022), GRU (Sutskever et al.) and Transformer (Vaswani et al.). Among them, Transformer has become popular recently due to its potential to capture long-term sequences. However, the problem to be solved in this study is the resource limitation of intelligent edge devices, and the computational complexity of Transformer is $O(n^2)$ which leads us to choose MLP with lower computational complexity $O(n)$.

Our model employs an MLP architecture primarily because of its computational efficiency and lightweight nature, which are essential for real-time traffic prediction on resource-constrained devices. Unlike recurrent models such as LSTM or GRU, which are computationally expensive, our approach focuses on reducing the computational load while still capturing temporal dependencies effectively. To achieve this, we use a sliding window approach to directly encode historical traffic data into the input feature space (as detailed in Section 3.1). This method allows the model to treat temporal patterns as part of its static input features, thus avoiding the complexity of recurrent structures while still capturing relevant time-series information. Experimental results confirm that our model achieves accuracy comparable to recurrent models, demonstrating that MLP can effectively capture temporal dependencies without the associated computational cost.

As shown in Fig. 6, H_m^j and H_m^i represent node values of each hidden layer, and i and j represent the ith and jth hidden layers. For the input data $\mathcal{X} = (X_{t_1}, X_{t_2}, ..., X_{t_\tau}) \in \mathbb{R}^{\tau \times N \times C}$, which represents the temporal information of the node. We divide it

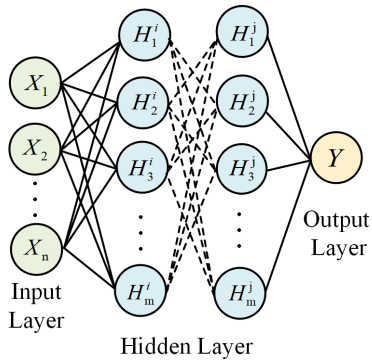


Fig. 6 Temporal model with MLP

into $N \times \mathcal{X}^{\mathcal{MLP}} = (X_{t_1}, X_{t_2}, ..., X_{t_\tau}) \in \mathbb{R}^{\tau \times C}$ as the input of the MLP. With $H^{(k)}$, $W^{(k)}$, and $b^{(k)}$ denoting the node values, parameter matrix and bias parameter of the kth hidden layer, respectively, we can obtain the output of the first hidden layer as follows:

$$H^{(1)} = \mathrm{Re}LU(X^{MLP} W^{(1)} + b^{(1)}) \in \mathbb{R}^{\tau \times C}. \quad (7)$$

The output of the lth hidden layer is as follows:

$$H^{(l)} = \mathrm{Re}LU(H^{(l-1)} W^{(l)} + b^{(l)}) \in \mathbb{R}^{\tau \times C}, \quad (8)$$

where $\mathrm{Re}LU(\cdot)$ represents the activation function:

$$\mathrm{Re}LU(x) = \max(x, 0). \quad (9)$$

Then we obtain the output of MLP Y:

$$Y^{MLP} = \mathrm{Re}LU(H^{(last-1)} W^{(last)} + b^{(last)}) \in \mathbb{R}^{\tau \times C}. \quad (10)$$

Finally, we concatenate all the N outputs:

$$Y^{Temporal} = Concat(Y_1^{MLP}, Y_2^{MLP}, ..., Y_n^{MLP}) \in \mathbb{R}^{\tau \times N \times C}. \quad (11)$$

4.4 Details of the OTL algorithm

As shown in Algorithm 1 and Fig. 7, we first feed and train the model with the classified data sets (as will be described in Section 5) to obtain the model convergence time and performance metrics $loss$, which includes mean absolute error (MAE), mean absolute error pecent (MAPE), and root mean square error (RMSE). All the pre-trained models correspond to Fig. 4 with different parameters.

Then, as shown in Algorithm 2, based on the signals collected by the intelligent edge device in real

Unedited

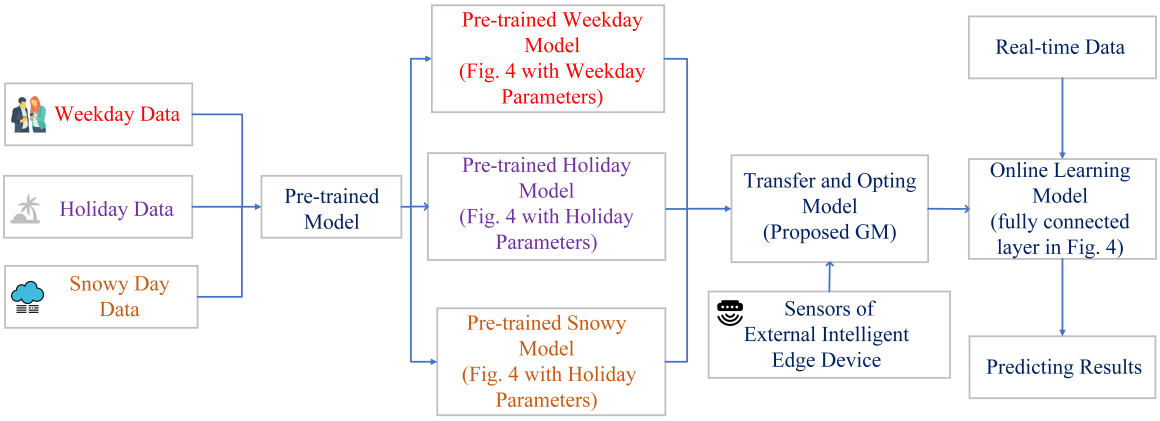


Fig. 7 OTL framework and data flow

Algorithm 1 Pre-trained Task

Input: $sourceLoader$, $sourceModel$, $criterion$, $optimizer$, $epochs$
Output: $convergenceTime$, $loss$
$startTime \leftarrow$
for $epoch \leftarrow 1$ **to** $epochs$ **do**
  for each $(inputs, labels) \in sourceLoader$ **do**
    $optimizer.zero_grad()$
    $outputs \leftarrow sourceModel(inputs)$
    $loss \leftarrow criterion(outputs, labels)$
    $loss.backward()$
    $optimizer.step()$
  end for
end for
$endTime \leftarrow$
$convergenceTime \leftarrow endTime - startTime$
return $convergenceTime$, $loss$

time, the device calculates the Euclidean distance between the collected data type and the pre-trained model. A pre-trained model corresponding to the smallest Euclidean distance to transfer is then selected. This step corresponds to the Pre-trained Model to Transfer and Opting Model in Fig. 7. After the transferred model is loaded into the current model, we freeze the parameters of the feature extraction layers(i.e., all layers except the fully connected layer), which means online learning is used at the fully connected layer in Fig. 4. Then we adjust the learning rate of the fully connected layer and continuously run the online learning algorithm in the current environment to obtain the model convergence time and *loss*. This corresponds to the Real-time Data to Online Learning Model in Fig. 7.

Algorithm 2 Online Transfer Learning Task

Input: $targetLoader$, $transferedModel$, $epochs$ $criterion$, $optimizer$, $curModel$, $signal$
Output: $convergenceTime$, $loss$
$EuclideanDistance() \leftarrow signal$
$transferedModel \leftarrow smallestED$
$curModel.loadStateDict($
  $transferedModel.stateDict())$
$parametersFreezingExceptFinalLayer$
$optim.Adam(finalLayer.parameters())$
$startTime \leftarrow$
for $epoch \leftarrow 1$ **to** $epochs$ **do**
  for each $(inputs, labels) \in tragetLoader$ **do**
    $optimizer.zero_grad()$
    $outputs \leftarrow curModel(inputs)$
    $loss \leftarrow criterion(outputs, labels)$
    $loss.backward()$
    $optimizer.step()$
  end for
end for
$endTime \leftarrow$
$convergenceTime \leftarrow endTime - startTime$
return $ConvergenceTime$, $loss$

The $startTime \leftarrow$ and $endTime \leftarrow$ represent obtaining the starting/ending time before/after the algorithm converges. Due to space limitations, we will write the smallest Euclidean distance as smallestED. lr is the learning rate.

Unedited

5 Experiments

5.1 Dataset

- Traffic Data: The datasets are collected by the Caltrans Performance Measurement System (PeMS) in real time every 30 s. The traffic data are aggregated into every 5-min intervasl from the raw data. This study verified the effectiveness of OTL-GM on the PeMS04 dataset provided by Los Angeles, California, USA. PeMS continuously collected data using sensors for 59 days from January 1, 2018, which contains data from 307 nodes. Sensors collected data every 5 min, so there are 16,992 records in total. Three-dimensional data features are flow, occupancy, and speed. According to the visualization results, this study adopted flow for performing experiments. We divided flow into two parts, 45 days for training and 14 days for testing. Based on the above, we started to reorganize PeMS04 into classified datasets.

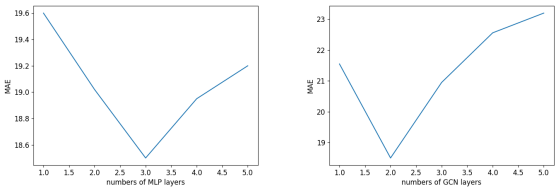
- The date type: We determined the weekend and weekdays for the 59 days using the Google Calendar (`https://calendar.google.com`). We separated the PeMS04 dataset into weekday and weekend datasets. There are 16 weekend days and 43 working days.

- The weather type: Based on the National Weather Service (`https://www.weather.gov`), entering the downtown area of Los Angeles, we identified the actual snowy days during the 59 days as, January 25, February 18, 19, 20, 26, and 27.

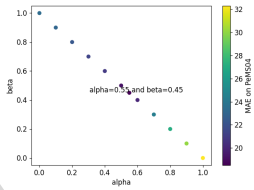
For ablation experiments, when the OTL algorithm was not used, the original data set was disrupted on a daily basis, and a dataset with the same sample length as the classified dataset using the OTL algorithm.

5.2 Experiment setting

The experiment was carried out in the following hardware configuration. As shown in Fig. **??**, the edge intelligent device NVIDIA JETSON NANO B01 is equipped with an ARM (R) Quad-core A57 CPU @ 1.43 GHz and 128 core Maxwell GPU 4GB 64-bit LPDDR4. We adopted the MSE function as the loss function and the Adam optimizer. The



(a) MLP (b) GCN



(c) Weight of Temporal and Spatial

Fig. 8 Parameter adjustment process on the validation set

hyper-parameter settings were as follows: To ensure the efficiency of model training, the learning rate of the Adam optimizer was 0.01, the batch size of the input data was 64, and the training epoch was based on the convergence time of various data sets. The hyper-parameters involved in OTL-GM included the number of MLP layers Num_M, the number of GCN layers Num_G, and the weight of spatio features α and temporal features β. These parameters were adjusted on the validation set to achieve the best performance of the OTL-GM ($Num_M = 3$, $Num_G = 2$, $\beta = 0.45$, $\alpha = 0.55$).

As shown in Fig. 8, the best performance is achieved the there are three MLP layers and two GCN layers; the performance is best when the spatial and temporal feature weights α and β are 0.55 and 0.45, respectively.

5.3 Evaluation metrics

Because traffic flow forecasting is essentially a regression problem, we use the classical indicators of regression problems to evaluate the effectiveness of the proposed method. They are MAE, MAPE, and RMSE, which are defined as:

$$MAE = \frac{1}{TS}\sum_{j=1}^{S}\sum_{i=1}^{T}\left|y_{ij} - x_{ij}\right|, \tag{12}$$

$$MAPE = \frac{1}{TS}\sum_{j=1}^{S}\sum_{i=1}^{T}\left|\frac{y_{ij} - x_{ij}}{y_{ij}}\right|, \tag{13}$$

Unedited

$$RMSE = \sqrt{\frac{\sum_{j=1}^{S}\sum_{i=1}^{T}(y_{ij}-x_{ij})^2}{TS}}\,, \tag{14}$$

where x_{ij} is the ground truth, y_{ij} is the prediction results. T represents the time dimension, and S is the spatial dimension.

The metric $Time$ records the total time from training to convergence of the model:

$$Time = Time_{end} - Time_{start}\,, \tag{15}$$

where $Time_{start}$ stands for the real time before training and $Time_{end}$ represents the real time after convergence.

5.4 Ablation study of the proposed GM

In this section, we describe a comprehensive ablation experiment to verify the performance of the GM. This does not involve the comparison of algorithm usage for OTL. The following baselines are widely used for time series forecasting or proposed in recent years and all hyper-parameters have been tuned to their optimal values:

- **HA**: This baseline uses the average of historical observations to estimate future observations.
- **Vector auto-regression (VAR)**: This baseline is a time series model designed to capture spatial correlations among various traffic series. (Zivot and Wang, 2006)
- **DCRNN**: This baseline integrates graph convolution with the diffusion process and combines GCNs with recurrent models in an encoder-decoder fashion for multi-step prediction. (Li et al., 2018)
- **STGCN**: This baseline is a spatio-temporal graph convolutional network that utilizes Graph Convolutional Networks (GCN) and temporal convolution to capture spatial and temporal correlations, respectively. (Yu et al.)
- **AGCRN**: This baseline is designed to capture fine-grained spatial and temporal correlations in traffic series automatically based on the two modules and recurrent networks. (Bai et al., 2020)
- **Cy2Mixer**: This baseline is a novel spatiotemporal GNN based on topological non-trivial invariants of spatio-temporal graphs with MLPs. (Lee et al., 2024)
- **STAEformer**: This baseline can yield outstanding results with vanilla transformers. (Liu et al., 2023)

The experimental results of MAE, MAPE, and RMSE over 12 predictive steps on the PeMS04 dataset are shown in Table 3.

Table 3 Metrics of all methods on PeMS04

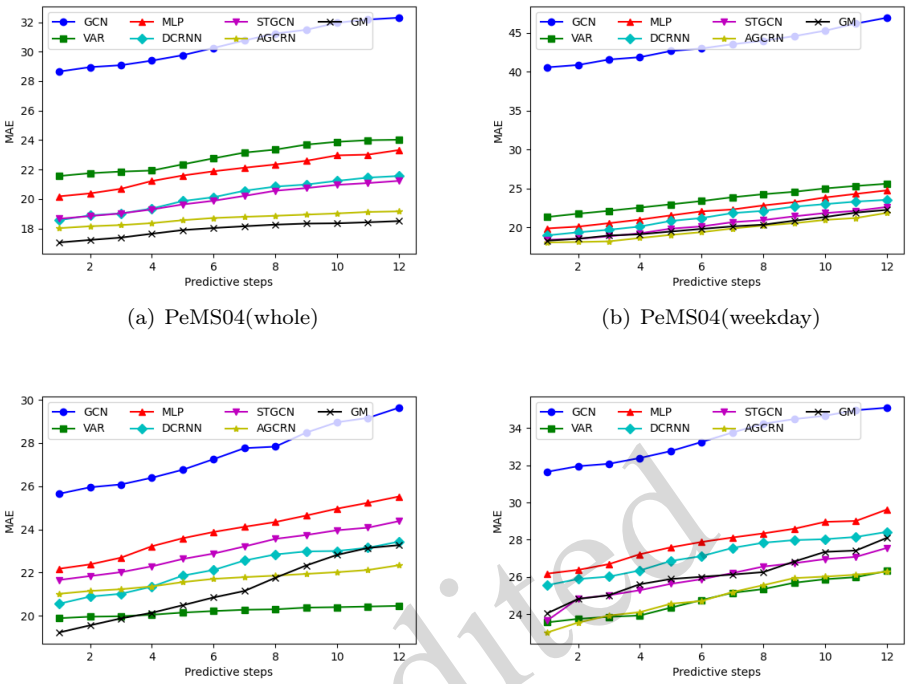
	MAE	MAPE (%)	RMSE
HA	38.85	28.22	58.86
GCN	32.30	32.40	46.08
VAR	24.02	17.55	37.21
MLP	23.32	15.52	35.07
DCRNN	21.56	14.88	34.85
STGCN	21.32	13.83	35.02
AGCRN	19.16	12.56	31.56
Cy2Mixer	**18.14**	**12.13**	28.22
STAEformer	18.22	12.85	28.85
GM	18.50	13.41	**27.95**

In Table 3, the bold values indicate the best performance achieved by the algorithms for the respective metrics. We can see that the deep learning method is better than the traditional model-driven method in terms of accuracy, and there is a little difference in convergence time between deep learning methods. As mentioned before, the accuracy of a model that only considers temporal or spatial characteristics is not as good as a model that considers both spatio-temporal characteristics, which is well reflected in GCN, MLP, and GM. This also explains why most studies now consider both temporal and spatial features. The results also show that on the dataset PeMS04, GM has reached the current research level, which enables the proposed GM to play a role in OTL.

We can see that the metrics of the proposed GM are not the best, but close to the best. We will explain the reason by combining Table 4 and Fig. 8.

As shown in Table 4, some of the latest papers have indeed improved in accuracy, but at the expense of algorithm complexity (discussed in Section 2.2) and algorithm convergence time. GCN, VAR, and MLP also reach low convergence time and parameters, but they fail to reach the current level of accuracy as mentioned above. However, our proposed GM considers the spatio-temporal correlation while simplifying the models and number of layers to

Unedited



(a) PeMS04(whole) (b) PeMS04(weekday)

(c) PeMS04(holiday) (d) PeMS04(weather)

Fig. 9 Accuracy comparison as the prediction horizon varies

Unedited

Table 4 Time and parameters of all methods on PeMS04

	Time (s)	Parameters
HA	125.58	-
GCN	134.58	9.12×10^4
VAR	152.25	9.86×10^4
MLP	155.55	2.85×10^5
DCRNN	164.42	3.88×10^5
STGCN	178.85	4.21×10^5
AGCRN	180.56	7.44×10^5
Cy2Mixer	280.43	6.28×10^6
STAEformer	240.98	4.37×10^6
GM	158.81	2.57×10^5

create a trade-off between accuracy and time complexity. As shown in Figs. 8(a) and (b), on one hand, increasing the model complexity can improve accuracy on the training set; on the other hand, overly complex models can lead to overfitting and increase the algorithm's convergence time, which contradicts our goal of a lightweight model.

5.5 Stability study of the proposed GM

As shown in Fig. 9, when the prediction horizon increases, the predictive accuracy of each method on every dataset decreases, which is consistent with the trends observed in deep learning. However, when the prediction horizon reaches 12 steps (1 h), our proposed model GM maintains a certain degree of superiority in predictive accuracy compared to other baseline models. This provides more time for effective path planning and navigation by IoT traffic managers. When the horizon reaches 1 step (5 min), all the accuracy improved and the proposed GM remains among the best models in PeMS04 (whole) and PeMS04 (holiday). Therefore, the proposed GM can achieve both short-term and long-term traffic flow prediction.

To further evaluate the training dynamics of the model, Fig. 10 shows the training and validation loss across epochs. This analysis provides insight into the potential risk of overfitting and the convergence behavior of the model. During training, we monitored the loss values for both training and validation datasets at each epoch. The loss function used was

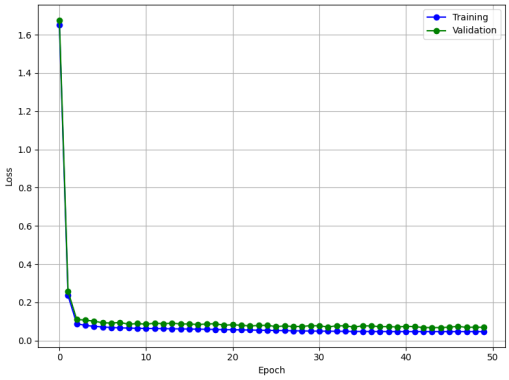


Fig. 10 Loss during training and validation across epochs

normalized MSE. The model converges quickly and the loss of the validation set is slightly higher than that of the training set.

5.6 Ablation study of proposed Transfer Learning

For ablation experiments in transfer learning, we compared the convergence time of algorithms using transfer learning and not using transfer learning under varying degrees of data sparsity. Taking data set PeMS04(weekday) with our proposed GM as an example, the results are shown in Table 5.

Table 5 Convergence time under varying degrees of data sparsity

Degrees	Time with TL (s)	Time without TL (s)
10%	4.41	60.54
20%	4.52	68.22
30%	4.93	75.91
40%	5.68	123.58
50%	6.72	*None*
60%	8.64	*None*
70%	11.56	*None*

We can see that algorithm convergence time with transfer learning slowly increases as data sparsity increases, while algorithm convergence time without transfer learning increases dramatically. More importantly, when the data sparsity reaches 50%, the algorithm without transfer learning no longer converges, which results in traffic flow predictions being unavailable.

However, one problem with transfer learning is that prediction accuracy decreases when the local data distribution changes significantly after migration. Therefore, we propose an online learning method to continuously update certain parameters of the model through real-time collected data to eliminate the decrease in accuracy caused by differences in data distribution.

5.7 Time and Space Complexity Analysis

Due to the use of adjacency matrix storage, the space complexity of the GCN is $O(N)$, where N represents each node in the actual transportation network and the time complexity of each layer of the GCN is $O(N)$, so it is still $O(N)$ after ignoring the constants. The space and time complexities of a VAR model primarily depend on the number of time steps T, the number of nodes N, and the lag order P.In particular, the time complexity is $O(T^2N^2P^2)$ and the space complexity is $O(PN^2 + TN)$. Because P and T are treated as constants, the final time complexity is $O(N^2)$ and space time is $O(N^2)$. Because the MLP is a time model, the time and space complexity is related to the input data scale, so in this article the time and space complexity is $O(1)$. The DCRNN consists of Diffusion Convolution and an RNN. After ignoring constants and lower-order variables, the space complexity of the DCRNN is $O(N^2)$ and the time complexity is $O(N)$. The STGCN consists of a GCN and GRU; and the time complexity is $O(N)$ and the space complexity is $O(N^2)$. The AGCRN consists of a GCN, Adaptive Graph Convolution, and an RNN. The time complexity is $O(N)$ and the space complexity is $O(N^2)$. Cy2Mixer consists of a GCN and Mixer attention; the time complexity is $O(N)$ and the space complexity is $O(N^2)$. STAEformer consists of Self-Attention, Spatio-Temporal Self-Attention, Transformer, and Position Coding. The space and time complexity are both $O(N^2)$. Our proposed GM uses a sparse matrix to store traffic nodes, so the space complexity is $O(N)$. And the MLP have been discussed above, so the time complexity is $O(1)$.

According to the above analysis, the GM offers significant advantages in handling resource-limited issues like traffic flow prediction on edge intelligent devices due to their efficient space and time complexities.

Unedited

Table 6 Performance and inference time comparison

Methods	MAE	MAPE (%)	RMSE	Time (s)
ST-GFSL	18.41	16.56	29.13	57.43
ST-DAAN	18.33	15.15	28.00	65.43
OTL-GM	18.38	15.14	28.18	2.20

5.8 Comparative Analysis with Transfer Learning Approaches

To further evaluate our method, we conducted experiments comparing it with two state-of-the-art transfer learning techniques:

- **ST-GFSL**: This technique proposes to generate non-shared parameters based on node-level meta knowledge (Lu et al., 2022).
- **ST-DAAN**: This technique maps the raw spatio-temporal data of the source domain and then employs domain adaptation (Wang et al., 2022).

The experiments were conducted on the PeMS04 (holiday) dataset, using the same training/test split and evaluation metrics. Table 6 summarizes the results and shows that our method achieves comparable performance while maintaining lower convergence time, that is, lower computational cost.

The results demonstrate that while transfer learning methods like (Lu et al., 2022) and (Wang et al., 2022) achieve high prediction accuracy, they require substantial computational resources. (Lu et al., 2022) involves weighted matrix calculation parameters and reconstructed graphs and (Wang et al., 2022) involves 3D convolution. Our approach, on the other hand, achieves a balance between accuracy and efficiency by just calculating Euclidean distance and adjusting the last fully connected layers, making it more practical for real-world deployment in resource-constrained scenarios.

5.9 Ablation study of the proposed OTL

In the previously mentioned transfer learning, the problem of data sparsity is improved. However, if the algorithm only uses transfer learning, the accuracy of predictions will decrease when the distribution of data changes significantly over time. Therefore, it is necessary to use online learning to continuously collect real-time data to update certain parameters of the model and maintain prediction accuracy.

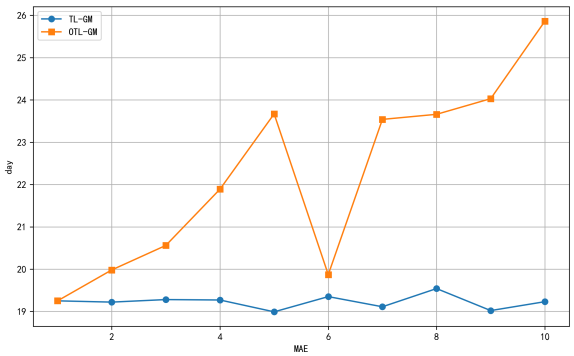


Fig. 11 Comparison between transfer learning and OTL

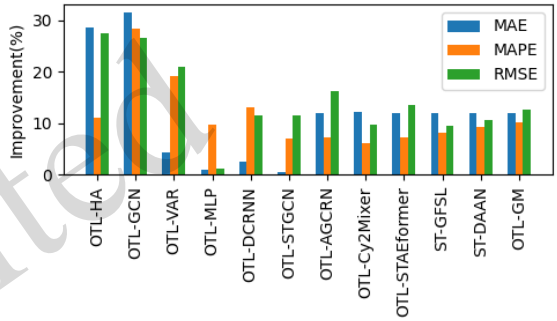


Fig. 12 Improvement of metrics on the PeMS04 (weekday) dataset

This part of the experiment was completed when the data sparsity was 30%. We still use the GM we propose to conduct experiments on the PeMS04 (weekday) dataset. The results are shown in Fig. 11.

As time passes and the data distribution changes continuously, solely relying on transfer learning can lead to a gradual decrease in prediction accuracy in the overall trend. However, the incorporation of online learning helps maintain the accuracy at a consistent level. It is worth noting that on the sixth day, the accuracy of transfer learning is close to that of the first day, which is due to the very small Euclidean distance between the data distribution on the sixth day and the first day.

5.10 All results of methods with OTL

As shown in Figs. 12-14 and Table 7, to improve the prediction metrics (MAE, MAPE and RMSE), the analysis of all the methods and datasets are as follows (the data sparsity is 30%):

- **PeMS04 (weekday)**: As shown in Fig. 12,

Unedited

Table 7 Metrics of all methods on the weekday, holiday and weather of PeMS04

	PeMS04 (weekday)				PeMS04 (holiday)				PeMS04 (weather)			
	MAE	MAPE (%)	RMSE	Time (s)	MAE	MAPE (%)	RMSE	Time (s)	MAE	MAPE (%)	RMSE	Time (s)
HA	48.42	38.55	65.43	90.35	33.57	27.99	53.11	48.85	40.52	48.23	60.23	26.16
GCN	46.92	36.75	62.69	96.90	29.63	32.40	42.29	32.33	35.09	27.73	49.77	19.17
VAR	25.62	21.58	38.02	89.26	20.46	17.25	31.18	48.37	26.32	27.47	39.22	20.63
MLP	24.77	18.36	36.28	69.38	25.52	18.60	38.29	21.48	29.62	36.57	39.03	12.14
DCRNN	23.56	16.28	35.62	68.58	23.43	17.43	37.24	42.24	28.42	32.46	36.12	18.33
STGCN	22.64	15.08	36.64	70.58	24.38	16.42	38.25	40.25	27.56	31.45	36.14	19.58
AGCRN	21.88	13.43	35.21	72.33	22.34	15.52	34.26	50.25	26.28	29.43	32.48	20.69
Cy2Mixer	21.74	13.08	33.44	123.56	21.85	15.18	33.18	80.33	25.98	30.14	38.23	25.48
STAEformer	21.78	12.96	34.02	111.35	21.98	15.27	34.01	76.48	26.03	29.00	38.88	23.19
GM	22.27	15.69	33.28	75.91	23.27	18.97	33.07	39.25	34.60	35.93	49.74	19.37
OTL-HA	34.52	34.23	47.44	14.02	32.54	27.93	50.85	5.56	37.49	45.72	55.42	14.02
OTL-GCN	32.12	26.27	45.96	13.00	29.25	32.21	41.73	4.12	33.28	26.01	47.42	13.28
OTL-VAR	24.52	17.43	30.02	15.52	20.12	17.02	30.25	5.02	23.98	26.17	37.85	15.52
OTL-MLP	24.55	16.58	35.86	16.01	25.23	18.45	34.74	5.93	26.77	31.08	36.23	2.79
OTL-DCRNN	22.98	14.18	31.52	9.65	23.01	17.21	35.98	3.58	26.47	32.18	34.12	8.32
OTL-STGCN	22.53	14.02	32.28	7.56	23.95	15.89	37.52	3.99	25.74	27.93	34.38	6.78
OTL-AGCRN	19.24	12.44	29.51	5.43	19.25	15.03	29.21	2.35	23.58	21.49	**31.51**	4.3
OTL-Cy2Mixer	**19.10**	12.28	30.18	28.12	**18.14**	14.33	28.20	13.34	**23.33**	22.65	37.78	18.63
OTL-STAEformer	19.19	**12.01**	29.41	20.55	18.23	**14.02**	29.01	10.56	23.46	**20.38**	37.62	17.32
ST-GFSL	19.23	13.89	29.34	113.35	18.41	16.56	29.13	57.43	23.38	24.51	36.47	68.43
ST-DAAN	19.14	13.31	29.28	98.72	18.33	15.15	**28.00**	65.43	23.35	23.31	34.58	80.72
OTL-GM	19.25	14.09	**29.04**	**4.93**	18.38	15.14	28.18	**2.20**	25.54	26.09	36.26	**1.10**

Unedited

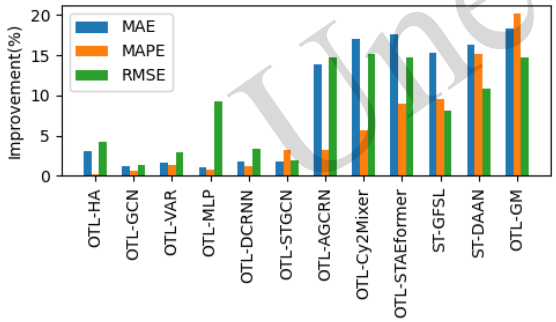


Fig. 13 Improvement of metrics on PeMS04 (holiday)

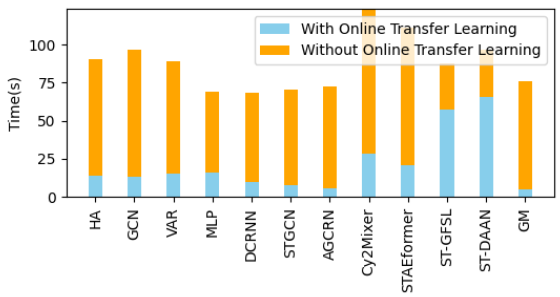


Fig. 15 Comparison of convergence times of all methods on PeMS04 (weekday)

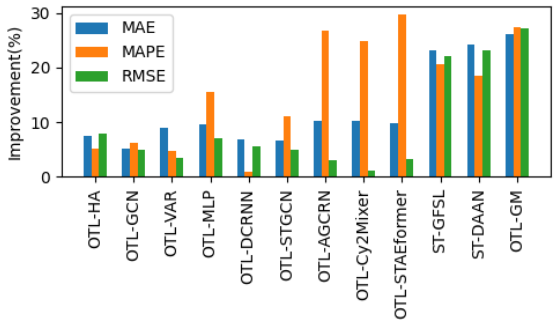


Fig. 14 Improvement of metrics on PeMS04 (weather)

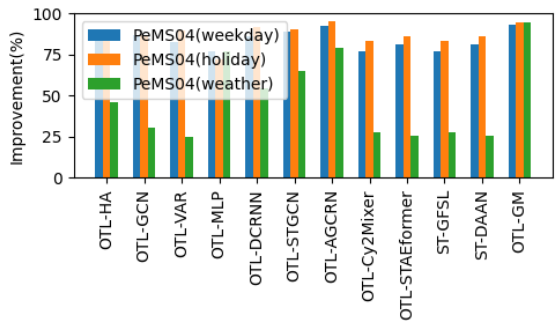


Fig. 16 Improvement of convergence time on all the classified datasets

the accuracy metrics of each method have been improved to a certain extent. For the MAE, the accuracy has increased from 0.48% (OTL-STGCN) to 31.54% (OTL-GCN); for the MAPE, the accuracy has increased from 7.03% (OTL-STGCN) to 28.51% (OTL-GCN); and for the RMSE index, the accuracy has increased from 1.16% (OTL-MLP) to 27.50% (OTL-HA). The proposed method OTL-GM has improved by 13.56%, 10.20% and 12.74%, respectively.

- **PeMS04 (holiday)**: As shown in Fig. 13, the results are similar to the above. For the MAE, the accuracy has increased from 1.13% (OTL-MLP) to 18.26% (OTL-GM); for the MAPE, the accuracy has increased from 0.21% (OTL-HA) to 20.19% (OTL-GM); and for the RMSE index, the accuracy has increased from 1.32% (OTL-GCN) to 15.10% (OTL-Cy2Mixer). The proposed method OTL-GM has improved by 18.26%, 20.19% and 14.78%, respectively.

- **PeMS04 (weather)**: As shown in Fig. 14, the results are similar to the above as well. For the MAE, the accuracy has increased from 5.16% (OTL-GCN) to 26.18% (OTL-GM); for the MAPE, the accuracy has increased from 0.86% (OTL-CGN) to 29.72% (OTL-STAEformer); and for the RMSE index, the accuracy has increased from 3.00% (OTL-AGCRN) to 27.10% (OTL-GM). The proposed method OTL-GM has improved by 26.18%, 27.38% and 27.10%, respectively.

Compared to the entire PeMS04 dataset, the sample size of three classified datasets decreased (43 days, 16 days, and 6 days, respectively), and the accuracy of all algorithms decreased, which is in line with the rules of deep learning. Based on the above results, we can conclude that, compared to merging heterogeneous datasets with chaotic distributions, utilizing OTL algorithms to merge datasets with similar data distributions, that is, the smaller Euclidean distance, has led to a certain degree of improvement in prediction accuracy, which makes the prediction results valid. The most important point is that our proposed OTL-GM achieves the best performance in terms of algorithm convergence time.

Taking Fig. 15 as an example, the convergence time of all methods on PeMS04 (weekday) has imoroved considerably. Fig. 16 presents the convergence time of all algorithms; the analysis of all the datasets is as follows:

- **PeMS04 (weekday)**: The convergence time of all algorithms has been significantly improved, increasing from 76.92% (OTL-MLP) to 93.51% (OTL-GM).

- **PeMS04 (holiday)**: The convergence time of all algorithms has seen a substantial enhancement, progressing from 72.40% (OTL-MLP) to 95.32% (OTL-AGCRN).

- **PeMS04 (weather)**: The convergence time for all algorithms has experienced a substantial improvement, increasing from 24.77% (OTL-VAR) to 94.32% (OTL-GM).

Through an analysis combining predictive accuracy and algorithm convergence time, we observe that algorithms adopting OTL not only enhance accuracy due to the similarity in learned data distributions but also significantly reduce convergence time, which can be attributed to a pretrained tasks algorithm.

As mentioned previously, many recent academic papers on traffic flow prediction focus on improving accuracy, which is often achieved by introducing complex components (Transformer) or increasing algorithm complexity. Although accuracy is undoubtedly crucial for traffic flow prediction, the practical scenarios need to take the limited device resources into account. The model and methods we propose aim to maintain accuracy at the current research level while, to some extent, reducing algorithm convergence time and increasing the practicality of traffic forecasting.

Although we perform the pre-trained tasks on our intelligent devices (as shown in Fig. **??**), in the real-world scenario, pre-training on the edge servers with rich computing resources is a very good choice. The OTL with smaller Euclidean distance addresses, to some extent, the issue of data sparsity resulting from extreme weather conditions, and the shortened algorithm convergence time simultaneously addresses limitations in computational resources on intelligent edge devices. Hence, we have addressed the issues of data sparsity and limited computational resources as initially posed in the introduction, rendering traffic flow prediction more practically meaning-

Unedited

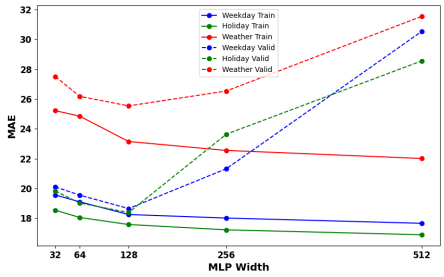


Fig. 17 Variation of MAE with different MLP widths

ful and closer to real-world applicability. The OTL method we propose can not only be applied to the GM, but to future advanced models proposed thereafter.

5.11 Sensitivity analysis to the MLP width

As mentioned in Section 5.2, the optimal depths for GCN and MLP are 2 and 3, and the width of the GCN is the number of traffic nodes, so this section will discuss the sensitivity of MLP width.

Fig. 17 shows that as the MLP width increases, the results on the training set gradually improve. However, for the PeMS04 (weekday) and PeMS04 (holiday) datasets, when the MLP width exceeds 128, the results on the test set begin to increase, indicating overfitting. In contrast, for the PeMS04 (weather) dataset, because we only collected a small amount of data (only 6 days), overfitting occurs when the MLP width exceeds 64. With sufficient data, our MLP model can generalize well to traffic flow prediction under different conditions.

5.12 Analysis of scalability and performance of the GM under operational conditions.

In real-world applications, scalability is a crucial aspect of deploying the OTLGM framework. As traffic increases and the time steps increase, significant computational challenges may be introduced. However, the OTL-GM framework is designed to handle such scenarios efficiently.

As shown in Table 7, the data volume of the three data sets PeMS04 (weekday), PeMS04 (holiday), and PeMS04 (weather) gradually increases. Judging from the time growth, from small to large, they are 1.10 s, 2.20 s and 4.93 s respectively. The amount of data in PeMS04 (weekday) is 7 times that of PeMS04 (weather), but the time has only increased by about 4 times. So the GM is well suited for a transportation network with increasing data volume. As shown in Fig. 9, when the step size increases, the accuracy gradually decreases, but it is still better than most methods and maintains a current level.

Addressing data sparsity is one of the goals of this study. We explored the convergence time of the GM algorithm under different data sparsity conditions. Table 5 shows that as data sparsity increases, algorithm convergence time also increases. When the data sparsity reaches 50%, the GM is no longer able to converge. However, transfer learning can successfully compensate for the sparseness of real-time data.

6 Conclusions

In this paper, within the context of IoT development, we addressed the challenges of traffic flow prediction in the real world by highlighting two issues: the limited computational resources of intelligent edge devices and the data sparsity resulting from extreme weather conditions. Utilizing Euclidean distance, we verified the similarity in distribution of the same type of data, which confirms the effectiveness of transfer learning. Considering the constraint of limited computational resources, we selected GCN models capable of capturing spatial features effectively and MLP models with lower complexity. We proposed a novel OTL algorithmic framework, integrating our proposed models and methods. Extensive validation experiments and ablation studies on the real-world PeMS04 dataset confirm the effectiveness of our proposed OTL-GM, addressing the limitations of computational resources and data sparsity on intelligent edge devices. This enhances the practical significance of traffic flow prediction.

We could not gather enough data under extreme weather conditions, which affected the accuracy of our predictions to some extent. However, our experiments show that the OTL approach we propose still works well. Despite this, ablation studies indicate that the proposed OTL approach remains effective. If a diverse set of data samples is available in sufficient quantity, it has the potential to improve prediction accuracy and reduce algorithm convergence time. We performed experiments only on our intelligent device and utilized OTL in real-world scenarios.

Unedited

Pretraining on edge servers with rich computing resources is a wise choice when combined with IoT technology. In the future, it will be a challenge to consider multiple types of data distributions simultaneously to achieve better prediction accuracy.

Contributors

Jingru SUN guided the research. Hongbo JIANG and Zhu XIAO processed the data. Yichuang SUN helped organize the paper. Chendingying LU drafted the paper and revised the final paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

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

Unedited

References

Asim M, Wang Y, Wang KZ, et al., 2020. A review on computational intelligence techniques in cloud and edge computing. *IEEE Trans Emerg Top Comput Intell*, 4(6):742-763. https://doi.org/10.1109/TETCI.2020.3007905

Bai L, Yao LN, Li C, et al., 2020. Adaptive graph convolutional recurrent network for traffic forecasting. proc 34th int conf neural information processing systems advances in neural information processing systems, article 1494.

Bando M, Hasebe K, Nakayama A, et al., 1995. Dynamical model of traffic congestion and numerical simulation. *Phys Rev E*, 51(2):1035-1042. https://doi.org/10.1103/PhysRevE.51.1035

Bojan TM, Kumar UR, Bojan VM, 2014. An internet of things based intelligent transportation system. Proc IEEE Int Conf Vehicular Electronics and Safety, p.174-179. https://doi.org/10.1109/ICVES.2014.7063743

Bruna J, Zaremba W, Szlam A, et al., 2014. Spectral networks and locally connected networks on graphs. Proc 2nd Int Conf Learning Representations, p.1-14.

Chen C, Li KL, Teo SG, et al., 2020. Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks. *ACM Trans Knowl Discov Data*, 14(4):42. https://doi.org/10.1145/3385414

Cong PJ, Zhou JL, Li LY, et al., 2021. A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud. *ACM Comput Surv*, 53(2):38. https://doi.org/10.1145/3378935

Crammer K, Dekel O, Keshet J, et al., 2006. Online passive-aggressive algorithms. *J Mach Learn Res*, 7(3):515-585.

Datla S, Sharma S, 2008. Impact of cold and snow on temporal and spatial variations of highway traffic volumes. *J Transp Geogr*, 16(5):358-372. https://doi.org/10.1016/j.jtrangeo.2007.12.003

Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. Proc 30th Int Conf on Neural Information Processing Systems, p.3844-3852.

Derawi M, Dalveren Y, Cheikh FA, 2020. Internet-of-things-based smart transportation systems for safer roads. Proc 6th World Forum on Internet of Things, p.1-4. https://doi.org/10.1109/WF-IoT48130.2020.9221208

Diao ZL, Wang X, Zhang DF, et al., 2019. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. Pro 33rd the AAAI Conf on Artificial Intelligence, p.890-897.

Gentile C, 2000. A new approximate maximal margin classification algorithm. Proc 14th Int Conf on Neural Information Processing Systems, p.479-485.

Gräber T, Lupberger S, Unterreiner M, et al., 2019. A hybrid approach to side-slip angle estimation with recurrent neural networks and kinematic vehicle models. *IEEE Trans Intell Veh*, 4(1):39-47. https://doi.org/10.1109/TIV.2018.2886687

Guo G, Yuan W, Liu JY, et al., 2023. Traffic forecasting via dilated temporal convolution with peak-sensitive loss. *IEEE Intell Transp Syst Mag*, 15(1):48-57. https://doi.org/10.1109/MITS.2021.3119869

Guo SN, Lin YF, Wan HY, et al., 2022. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Trans Knowl Data Eng*, 34(11):5415-5428. https://doi.org/10.1109/TKDE.2021.3056502

Hamed MM, Al-Masaeid HR, Said ZMB, 1995. Short-term prediction of traffic volume in urban arterials. *J Transp Eng*, 121(3):249-254. https://doi.org/10.1061/(ASCE)0733-947X(1995)121:3(249)

Han PC, Wang SQ, Leung KK, 2020. Adaptive gradient sparsification for efficient federated learning: An online learning approach. Proc 40th Int Conf on Distributed Computing Systems, p.300-310. https://doi.org/10.1109/ICDCS47774.2020.00026

Hashemi H, Abdelghany K, 2015. Real-time traffic network state prediction for proactive traffic management: Simulation experiments and sensitivity analysis. *Transp Res Rec*, 2491(1):22-31. https://doi.org/10.3141/2491-03

Jeong YS, Byon YJ, Castro-Neto MM, et al., 2013. Supervised weighting-online learning algorithm for short-term traffic flow prediction. *IEEE Trans Intell Transp Syst*, 14(4):1700-1707. https://doi.org/10.1109/TITS.2013.2267735

Jia YH, Wu JP, Du YM, 2016. Traffic speed prediction using deep learning method. Proc 19th international Conf on Intelligent Transportation Systems, p.1217-1222. https://doi.org/10.1109/ITSC.2016.7795712

Kashyap AA, Raviraj S, Devarakonda A, et al., 2022. Traffic flow prediction models–a review of deep learning techniques. *Cogent Eng*, 9(1):2010510. https://doi.org/10.1080/23311916.2021.2010510

Kipf TN, Welling M, 2017. Semi-supervised classification with graph convolutional networks. Proc 5th Int Conf on Learning Representations.

Krizhevsky A, Sutskever I, Hinton GE, 2017. Imagenet classification with deep convolutional neural networks. *Commun ACM*, 60(6):84-90. https://doi.org/10.1145/3065386

Kwon YD, Chauhan J, Mascolo C. Fasticarl: Fast incremental classifier and representation learning with efficient budget allocation in audio sensing applications. Proc 22nd Annual Conf of the International Speech Communication Association, p.498-504.

Lee M, Choi YY, Park SW, et al., 2024. A gated mlp architecture for learning topological dependencies in spatio-temporal graphs. https://arxiv.org/html/2401.15894v1

Li YG, Yu R, Shahabi C, et al., 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. Proc 6th Int Conf on Learning Representations.

Li Y, Long PM. The relaxed online maximum margin algorithm. Proc 13th Int Conf on Neural Information Processing Systems, p.498-504.

Liu HC, Dong Z, Jiang RH, et al., 2023. Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting. Proc 32nd ACM int Conf on information and knowledge management, p.4125-4129.

Liu J, Guan W. A summary of traffic flow forecasting methods. *J high transp res dev*, 21(3):82-85 (in Chinese). https://doi.org/10.3969/j.issn.1002-0268.2004.03.022

Lu B, Gan XY, Zhang WN, et al., 2022. Spatio-temporal graph few-shot learning with cross-city knowledge transfer. Proc 28th ACM SIGKDD Conf on Knowledge Discovery and Data Mining, p.1162-1172. https://doi.org/10.1145/3534678.3539281

Lv YS, Duan YJ, Kang WW, et al., 2014. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans Int Transp Syst*, 16(2):865-873. https://doi.org/10.1109/TITS.2014.2345663

Mallick T, Balaprakash P, Rask E, et al., 2021. Transfer learning with graph neural networks for short-term highway traffic forecasting. Proc 25th Int Conf on Pattern Recognition, p.10367-10374. https://doi.org/10.1109/ICPR48806.2021.9413270

Nellore K, Hancke GP, 2016. A survey on urban traffic management system using wireless sensor networks. *Sensors*, 16(2):157. https://doi.org/10.3390/s16020157

Qadri YA, Nauman A, Zikria YB, et al., 2020. The future of healthcare internet of things: a survey of emerging technologies. *IEEE Commun Surv Tutorials*, 22(2):1121-1167. https://doi.org/10.1109/COMST.2020.2973314

Rosenblatt F, 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol rev*, 65(6):386-408. https://doi.org/10.1037/h0042519

Rumelhart DE, Hinton GE, Williams RJ, 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533-536. https://doi.org/10.1038/323533a0

Singh R, Sharma R, Akram SV, et al., 2021. Highway 4.0: Digitalization of highways for vulnerable road safety development with intelligent iot sensors and machine learning. *Saf sci*, 143:105407. https://doi.org/10.1016/j.ssci.2021.105407

Sun JR, Peng M, Jiang HB, et al., 2022. Hmian: A hierarchical mapping and interactive attention data fusion network for traffic forecasting. *IEEE Internet Things J*, 9(24):25685-25697. https://doi.org/10.1109/JIOT.2022.3196461

Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Proc 28th Int Conf on Neural Information Processing Systems, p.3104-3112.

Tang C, Sun JR, Sun YC, et al., 2020. A general traffic flow prediction approach based on spatial-temporal graph attention. *IEEE Access*, 8:153731-153741. https://doi.org/10.1109/ACCESS.2020.3018452

Tzeng E, Hoffman J, Darrell T, et al., 2015. Simultaneous deep transfer across domains and tasks. Proc IEEE int conf on computer vision, p.4068-4076.

Van Lint H, Van Hinsbergen C, 2012. Short-term traffic and travel time prediction models. *Artif Intell Appl Crit Transp Issues*, 22(E-C168):22-41.

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Proc 31st Int Conf on Neural Information Processing Systems, p.6000-6010.

Wang SZ, Miao H, Li JY, et al., 2022. Spatio-temporal knowledge transfer for urban crowd flow prediction via deep attentive adaptation networks. *IEEE Trans Intell Transp Syst*, 23(5):4695-4705. https://doi.org/10.1109/tits.2021.3055207

Williams BM, Hoel LA, 2003. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *J transp eng*, 129(6):664-672. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:6(664)

Yao ZX, Xia SC, Li Y, et al., 2023. Transfer learning with spatial–temporal graph convolutional network for traffic prediction. *IEEE Trans Intell Transp Syst*, . https://doi.org/10.1109/TITS.2023.3250424

Yu B, Yin HT, Zhu ZX. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. Proc 27th Int Joint Conf on Artificial Intelligence.

Zhang CY, Patras P. Long-term mobile traffic forecasting using deep spatio-temporal neural networks. Proc 18th ACM Int Symp on Mobile Ad Hoc Networking and Computing, p.231-240. https://doi.org/10.1145/3209582.3209606

Zhang H, Lu XX, 2020. Vehicle communication network in intelligent transportation system based on internet of things. *Comput Commun*, 160:799-806. https://doi.org/10.1016/j.comcom.2020.03.041

Zhang T, Guo G, 2022. Graph attention lstm: A spatiotemporal approach for traffic flow forecasting. *IEEE Intell Transp Syst Mag*, 14(2):190-196. https://doi.org/10.1109/MITS.2020.2990165

Zhao L, Song YJ, Zhang C, et al., 2020. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE trans intell transp syst*, 21(9):3848-3858. https://doi.org/10.1109/TITS.2019.2935152

Unedited

Zhao PL, Hoi SCH, Jin R, 2011. Double updating online learning. *J Mach Learn Res*, 12:1587-1615.

Zhu JW, Wang QJ, Tao C, et al., 2021. Ast-gcn: Attribute-augmented spatiotemporal graph convolutional network for traffic forecasting. *IEEE Access*, 9:35973-35983. https://doi.org/10.1109/ACCESS.2021.3062114

Zivot E, Wang JH, 2006. Vector autoregressive models for multivariate time series. *Modeling financial time series with S-PLUS®* . Springer, New York, p.385-429. https://doi.org/10.1007/978-0-387-32348-0_11

Unedited