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 prediction: a solution for edge intelligent devices*

Jingru SUN[†‡1,2], Chendingying LU[1], Yichuang SUN[3], Hongbo JIANG[1], Zhu XIAO[1]

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

[2]*Chongqing Research Institute, Hunan University, Chongqing 401120, China*

[3]*School of Engineering and Computer Science, University of Hertfordshire, Hatfield AL10 9AB, UK*

[†]E-mail: jt_sunjr@hnu.edu.cn

Received Dec. 12, 2024; Revision accepted Mar. 2, 2025; Crosschecked Aug. 29, 2025

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) framework with a multi-layer perceptron (MLP)-assisted graph convolutional network (GCN), termed OTL-GM, which consists of two parts: transferring source-domain features to edge devices and using online learning to bridge domain gaps. Experiments on four data sets demonstrate OTL's effectiveness; in a comparison with models not using OTL, the reduction in the convergence time of the OTL models ranges from 24.77% to 95.32%.

Key words: Online transfer learning; Traffic prediction; Intelligent edge devices
https://doi.org/10.1631/FITEE.2401059 **CLC number:** U491.1; TP18

1 Introduction

The integration of the Internet of Things (IoT) and 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 H and Lu, 2020). The significance of IoT lies in processing data with advanced techniques such as deep learning, 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, prediction 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 ITSs.

On 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

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 62171182), the Natural Science Foundation of Chongqing, the Chongqing Science and Technology Commission (No. CSTB2022NSCQ-MSX0770), the Science and Technology Plan Project of Hunan Provincial Department of Transportation, China (No. 202306), and the Hunan Emergency Management Science and Technology Project (No. yjtkjxm_202407)

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

magnetic and infrared technologies. Later, cameras were introduced to improve monitoring precision. 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 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 perform computation, but also transmit wireless data (Cong et al., 2021) and information on road condition detection (Singh et al., 2021). Therefore, how to reduce the computing resource consumption and provide as many computing resources as possible for other aspects has become an urgent problem. 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 microsimulation and macro-models began to attract attention (Bando et al., 1995). During the 2010s, there was growing 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., 2017), diffusion convolutional recurrent neural network (DCRNN) (Li YG et al., 2018), and graph attention method (Tang et al., 2020), all of which achieve excellent accuracy. This pursuit of higher accuracy often comes with an increase in algorithmic complexity and prediction time (Diao et al., 2019; Zhu et al., 2021; Guo SN et al., 2022).

The increasing complexity of algorithms conflicts with 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 5th generation mobile communication (5G) modules for intercommunication (Fig. 2). They can accurately ascertain the traffic flow and 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 going out. Such extreme weather reduces the accuracy of detection algorithms like YOLO, which leads to data sparsity. Similarly, different periods exhibit changes in data distribution (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 have become a challenge.

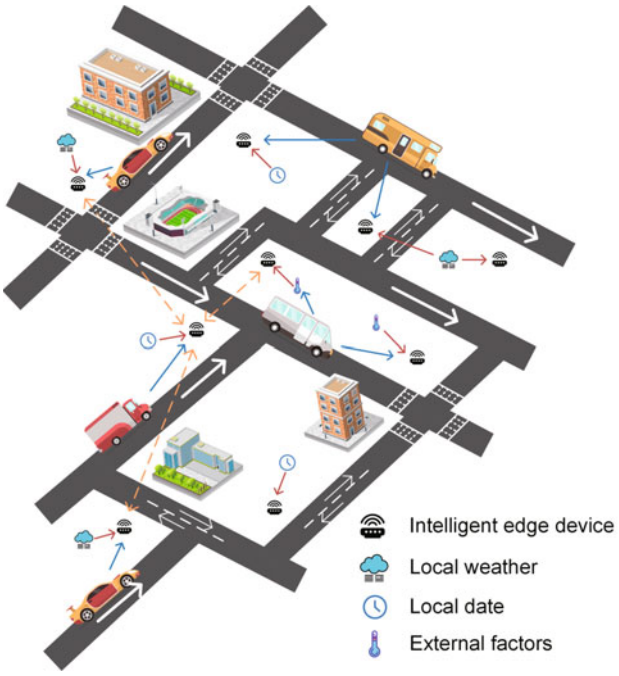

Fig. 1 A real-world scenario of traffic collection and prediction

Traditional traffic flow prediction algorithms often consider all features in the data set (Zhang CY and Patras, 2018), 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

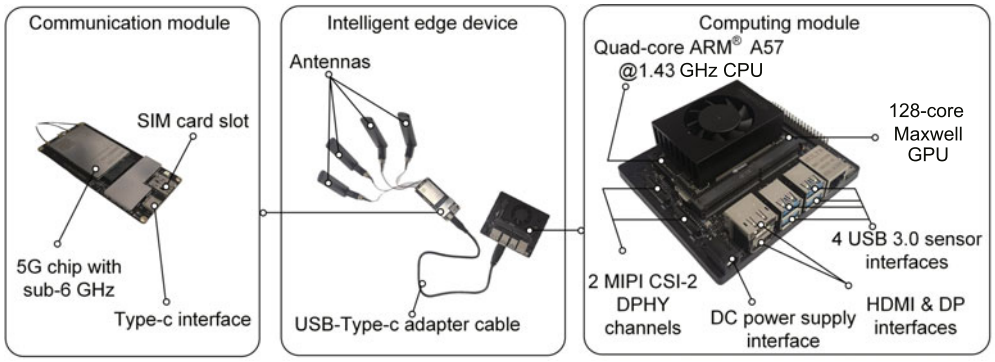

Fig. 2 Intelligent edge device with a 5G communication module and various interfaces

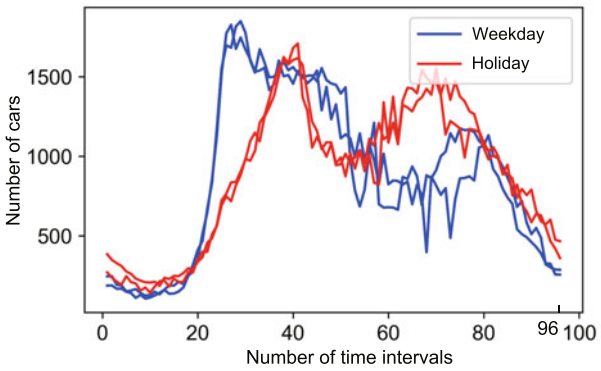

Fig. 3 Data distribution of weekdays and holidays. This data set collects real-time traffic data every 15 min, which means there are 96 collection points in a day. References to color refer to the online version of this figure

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. However, the literature is insufficient to address these emerging challenges in real-world scenarios. To solve them, we propose an online transfer learning (OTL) framework with a multi-layer perceptron (MLP)-assisted graph convolutional network (GCN), termed OTL-GM. The main contributions of this paper can be summarized as follows:

1. To ensure that the models running on intelligent edge devices are lightweight, we combine the temporal model MLP with a spatial model GCN to obtain a GM model that captures spatiotemporal features.

2. We combine the GM model with a novel OTL framework for traffic prediction, 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 distances.

3. We validate the effectiveness of our OTL framework by conducting extensive experiments on a real-world data set PeMS04, which is classified into three sub-data sets based on the data type. Algorithm convergence time has been greatly reduced. To the best of our knowledge, it is the first attempt to apply OTL for traffic flow prediction.

2 Related works

2.1 Traffic flow prediction

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 auto-regressive moving average model (Williams and Hoel, 2003), the history average (HA) model (Liu J and Guan, 2004), 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 attention in recent decades. Data-driven approaches include machine learning and deep learning ones. Machine learning approaches, in time

order, include k-nearest neighbors (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 the setting of the initial hyperparameters, that is, prior knowledge, which determines the accuracy of the prediction. 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 Spatiotemporal correlation

Given that traffic flow prediction is a time-series prediction problem, some neural networks for time-series prediction problems have been applied to traffic flow prediction, such as the recurrent neural network (RNN) (Gräber et al., 2019). The time complexity is $O(T)$, where T represents the length of the input sequence. Furthermore, to overcome the vanishing gradient problem and the exploding gradient of RNN, derivatives of RNN, such as the long short-term memory (LSTM) (Zhang T and Guo, 2022) neural network, started to emerge. Actually, the time complexity of LSTM is $O(4T)$ because of its three gates, but according to the rule of algorithm complexity, it is still $O(T)$. Gated recurrent unit (GRU) networks (Sutskever et al., 2014) 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 Transformer (Vaswani et al., 2017) at the cost of $O(T^2)$. Its parallelized self-attention mechanism enables efficient processing. However, the increased computational demands cannot meet the requirement for limited computational resources. We use an MLP and adjust 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 spatiotemporal 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) establishes multiple gated spatiotemporal CNN networks to extract multiple spatiotemporal dependencies and S-TCN (Guo G et al., 2023) is modeled with spatiotemporal 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 number of parameters based on the same adjacency matrix. Excellent work in recent years includes the primary GCN (Kipf and Welling, 2017), in which 2–3 layers lead to the best results, spatial–temporal GCN (Yu et al., 2017), which combines gated CNNs and a GCN, and the temporal GCN (Zhao L et al., 2020), which combines 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 spatiotemporal features.

2.3 Transfer learning

The motivation for adopting transfer learning in the 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 remain 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 similar model that has a similar data distribution. Nonetheless, transfer learning leads to prediction errors to some extent. Several studies have used different methods to solve this problem. Yao et al. (2023) proposed adversarial domain adaptation to reduce the error caused by transfer learning. Kwon et al. (2021) used 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 optimizing domain invariance to facilitate domain transfer and used a soft label distribution matching loss to transfer information between tasks. We use the Euclidean distance to characterize the similarity between data distributions and implement online 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 Y and Long, 1999; 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 adjusts the weight of the classifier based on the latest samples. Later, a new algorithm named double updating online learning (DUOL) emerged (Zhao PL et al., 2011), 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 used in the model (Table 1) and defines the task of the prediction model.

3.1 Traffic flow data

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

Table 1 Key notations

Symbol	Description
G	Graph of the road network
X	Data collected by the devices
$\boldsymbol{\mathcal{X}}$	A batch of collected data
$\mathcal{Y}$	Prediction results
τ	Length of $\boldsymbol{\mathcal{X}}$
λ	Prediction step size
C	Number of kinds of data
N	Number of nodes in the network
V	Set of nodes
E	Set of edges between any two nodes
$\boldsymbol{A}$	Adjacency matrix of G
T	Time dimension
S	Spatial dimension
$\hat{\boldsymbol{A}}$	Graph matrix with added self-connection
$\boldsymbol{I}_N$	Identity matrix
$\widetilde{\boldsymbol{D}}$	Degree matrix with added self-connection
$\boldsymbol{D}$	Degree matrix
$\boldsymbol{H}^{(l)}$	Output of the l^{th} layer
$\boldsymbol{\theta}^{(l)}$	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 $G = (V, E, \boldsymbol{A})$, where V represents all the nodes in the road network. Generally, the vertex, node, sensor, and detector are viewed as the same thing. For simplicity, we use node for the following description, which is denoted as $v_i \in V, i = 1, 2, ..., N$; E is the set of the edges in the whole network, indicating the edges between any two nodes; $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ denotes the adjacency matrix of graph G.

3.3 Definition of traffic prediction

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

problem is defined as follows:

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

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

3.4 Transfer learning

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

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

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 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 of data is much smaller than that of different types of data. Based on the results above and the real-time interaction of intelligent edge devices, we use transfer learning to obtain the pretrained models and choose the most suitable one. First, we train 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 obtain the signals (type of data and weather) from intelligent devices, and calculate the Euclidean distance between the data distribution of pretrained models and the data type represented by the signals. Finally, we choose the smallest one to transfer.

3.5 Online transfer learning

The transfer learning discussed in Section 3.4 involves only transferring a model with the most suitable data distribution. However, the distribution of real-time data may not align closely with that of the transferred model, so it is necessary for smart edge devices to continuously collect real-time data and update some parameters of the layers of the model to enhance prediction accuracy and timeliness.

4 Proposed model and algorithms

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

4.1 Overview of the GM framework

In this subsection, 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 $\boldsymbol{\mathcal{X}} = (\boldsymbol{X}_{t_1}, \boldsymbol{X}_{t_2}, ..., \boldsymbol{X}_{t_\tau}) \in \mathbb{R}^{\tau \times N \times C}$ is divided into small batches $\boldsymbol{\mathcal{X}}_B = (\boldsymbol{X}_{t_1}, \boldsymbol{X}_{t_2}, ..., \boldsymbol{X}_{t_B}) \in \mathbb{R}^{B \times t_B \times N \times C}$, with B and t_B representing the number of batches and the size of each batch respectively. $\boldsymbol{W}_{\text{Best-S}}$ and $\boldsymbol{W}_{\text{Best-T}}$ are the parameters of the best model transferred by our OTL framework. α and β represent the weights of time and space features, respectively. The final prediction result is $\boldsymbol{Y} = \alpha \boldsymbol{Y}^{\text{Spatial}} + \beta \boldsymbol{Y}^{\text{Temporal}}$. The details of each module and the OTL algorithm are described as follows.

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. 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., 2016) and image classification (Bruna et al., 2014). Observing the formula

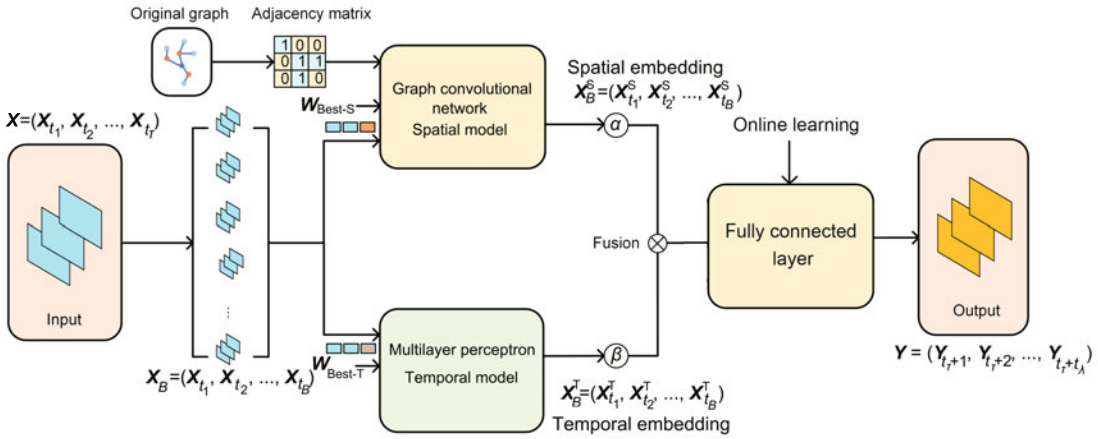

Fig. 4 The entire architecture of our proposed GM

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 and can extract spatial features between nodes by their first-order neighborhood. Guided by the prediction result, we can stack these filters to establish the GCN model, which is described as follows:

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

where $\hat{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}_N$ is the graph matrix with added self-connection, $\boldsymbol{A}$ is the adjacency matrix of the graph, and $\boldsymbol{I}_N$ is the identity matrix. $\widetilde{\boldsymbol{D}}=\boldsymbol{D}+\boldsymbol{I}_N$ is $\boldsymbol{D}$ with added self-connection, $\boldsymbol{D}=\sum_j \hat{\boldsymbol{A}}_{i,j}$ is the degree matrix of the graph, $\boldsymbol{H}^{(l)}$ is the output of the l^{th} layer, $\boldsymbol{\theta}^{(l)}$ represents the weight parameter corresponding to the l^{th} layer, and $\sigma(\cdot)$ is the sigmoid function for a nonlinear activation:

$$\sigma(x)=\frac{\exp(x)}{\exp(x)+1}. \tag{4}$$

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

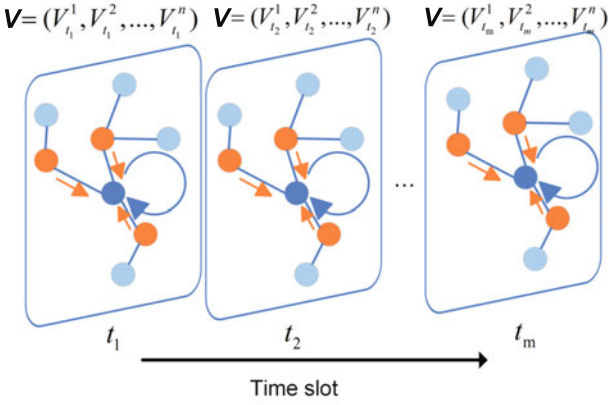

Fig. 5 The process of graph convolution aggregation (References to color refer to the online version of this figure)

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

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

and the output of GCN

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

4.3 MLP for the temporal feature

As mentioned previously, there are many excellent models for capturing temporal features, such as RNN, LSTM (Zhang T and Guo, 2022), GRU (Sutskever et al., 2014), and Transformer (Vaswani et al., 2017). 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 high computational efficiency and lightweight nature, which are essential for real-time traffic prediction on resource-constrained devices. Unlike recurrent models such as LSTM and GRU, which are computationally expensive, our approach focuses on reducing the computational load while 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 high complexity of recurrent structures while capturing relevant time-series information. Experimental results confirm that our model achieves accuracy comparable to that of recurrent models, demonstrating that MLP can effectively capture temporal dependencies without the associated computational cost.

As shown in Fig. 6, H_m^i and H_m^j represent the node values of each hidden layer, and i and j represent the i^{th} and j^{th} hidden layers respectively. For the input data $\boldsymbol{\mathcal{X}} = (\boldsymbol{X}_{t_1}, \boldsymbol{X}_{t_2}, ..., \boldsymbol{X}_{t_\tau}) \in \mathbb{R}^{\tau\times N\times C}$, which represents the temporal information of the node, we divide it into $N \times \boldsymbol{\mathcal{X}}^{\text{MLP}} = (\boldsymbol{X}_{t_1}, \boldsymbol{X}_{t_2}, ..., \boldsymbol{X}_{t_\tau}) \in \mathbb{R}^{\tau\times C}$ as the input of MLP. With $\boldsymbol{H}^{(k)}$, $\boldsymbol{W}^{(k)}$, and $\boldsymbol{b}^{(k)}$ denoting the node values, parameter matrix, and bias parameter of the k^{th} hidden layer, respectively, we can obtain the output of the first hidden layer as follows:

$$\boldsymbol{H}^{(1)} = \text{ReLU}\left(\boldsymbol{\mathcal{X}}^{\text{MLP}}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)}\right) \in \mathbb{R}^{\tau\times C}, \quad (7)$$

where $\text{ReLU}(\cdot)$ represents the activation function:

$$\text{ReLU}(x) = \max(x, 0). \quad (8)$$

The output of the l^{th} hidden layer is as follows:

$$\boldsymbol{H}^{(l)} = \text{ReLU}\left(\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)} + \boldsymbol{b}^{(l)}\right) \in \mathbb{R}^{\tau\times C}. \quad (9)$$

Then we obtain the output of MLP:

$$\boldsymbol{Y}^{\text{MLP}} = \text{ReLU}\left(\boldsymbol{H}^{(\text{last}-1)}\boldsymbol{W}^{(\text{last})} + \boldsymbol{b}^{(\text{last})}\right) \in \mathbb{R}^{\tau\times C}. \quad (10)$$

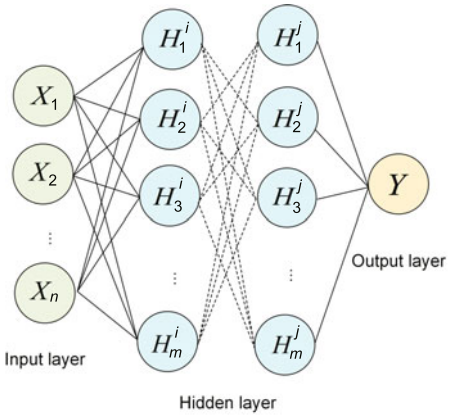

Fig. 6 The temporal model with MLP

Finally, we concatenate all the N outputs:

$$\boldsymbol{Y}^{\text{Temporal}} = \text{Concat}\left(\boldsymbol{Y}_1^{\text{MLP}}, \boldsymbol{Y}_2^{\text{MLP}}, \ldots, \boldsymbol{Y}_N^{\text{MLP}}\right) \in \mathbb{R}^{\tau\times N\times C}. \quad (11)$$

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 metric loss, including the mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean squared error (RMSE). All the pretrained 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 time, the device calculates the Euclidean distance between the collected data type and the pretrained model. A pretrained model corresponding to the smallest Euclidean distance to transfer is then selected. This step corresponds to the pretrained model to the 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 that 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 the online learning model in Fig. 7.

The startTime ← and endTime← represent obtaining the starting/ending time before/after the algorithm converges. Due to space limitation, we write

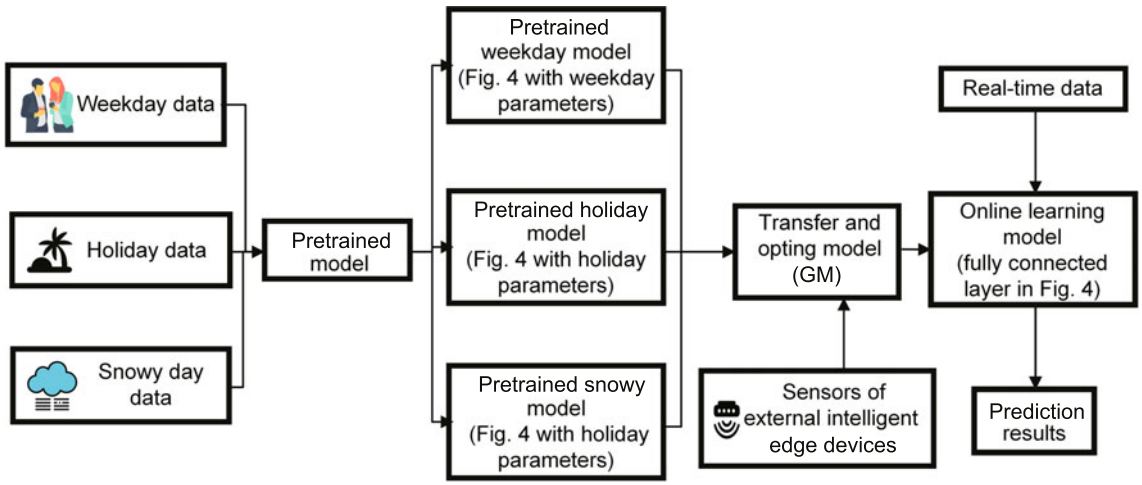

Fig. 7 The OTL framework and data flow

Algorithm 1 Pretrained task

Input: sourceLoader, sourceModel, criterion, optimizer, epochs
Output: convergenceTime, loss

```
startTime ← currentTime
for epoch = 1 to epochs do
    for each (inputs, labels) ∈ sourceLoader do
        optimizer.zero_grad()
        outputs ← sourceModel(inputs)
        loss ← criterion(outputs, labels)
        loss.backward()
        optimizer.step()
    end for
end for
endTime ← currentTime
convergenceTime ← endTime – startTime
return convergenceTime, loss
```

Algorithm 2 Online transfer learning task

Input: targetLoader, transferredModel, epochs, criterion, optimizer, curModel, signal
Output: convergenceTime, loss

```
EuclideanDistance() ← signal
transferredModel ← smallestED
curModel.loadStateDict(
      transferredModel.stateDict())
parametersFreezingExceptFinalLayer
optim.Adam(finalLayer.parameters())
startTime ← currentTime
for epoch = 1 to epochs do
    for each (inputs, labels) ∈ targetLoader do
        optimizer.zero_grad()
        outputs ← curModel(inputs)
        loss ← criterion(outputs, labels)
        loss.backward()
        optimizer.step()
    end for
end for
endTime ← currentTime
convergenceTime ← endTime – startTime
return convergenceTime, loss
```

the smallest Euclidean distance as smallestED. lr is the learning rate.

5 Experiments

5.1 Dataset

Traffic data: The data sets are collected by the Caltrans Performance Measurement System (PeMS) in real time every 30 s. The traffic data are aggregated into 5-min intervals from the raw data. In this study we verified the effectiveness of OTL-GM on the PeMS04 data set provided by Los Angeles, California, USA. PeMS continuously collected data using sensors for 59 d 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, we adopted flow for performing experiments. We divided the flow into two parts, 45 d for training and 14 d for testing. Based on the above, we started to reorganize PeMS04 into classified data sets.

Date type: We determined the weekends and weekdays for the 59 d using Google Calendar (https://calendar.google.com). We separated the PeMS04 data set into weekday and weekend data

sets. There are 16 weekends and 43 weekdays.

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 d 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 data set of the same sample length as the classified data set using the OTL algorithm.

5.2 Experiment setting

The experiment was carried out with the following hardware configuration. As shown in Fig. 2, the edge intelligent device NVIDIA JETSON NANO B01 was equipped with an ARM® Quad-core A57 CPU @ 1.43 GHz and 128-core Maxwell GPU 4-GB 64-bit LPDDR4. We adopted the MSE function as the loss function and the Adam optimizer. The hyperparameter 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 hyperparameters involved in OTL-GM included the number of MLP layers N_{M}, the number of GCN layers N_{G}, and the weights of spatial features α and temporal features β. These parameters were adjusted on the validation set to achieve the best performance of OTL-GM ($N_{\mathrm{M}}=3$, $N_{\mathrm{G}}=2$, $\beta=0.45$, $\alpha=0.55$).

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

5.3 Evaluation metrics

Because traffic flow prediction 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

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

$$\mathrm{MAPE}=\frac{1}{TS}\sum_{j=1}^{S}\sum_{i=1}^{T}\left|\frac{y_{ij}-x_{ij}}{y_{ij}}\right|\times 100\%, \tag{13}$$

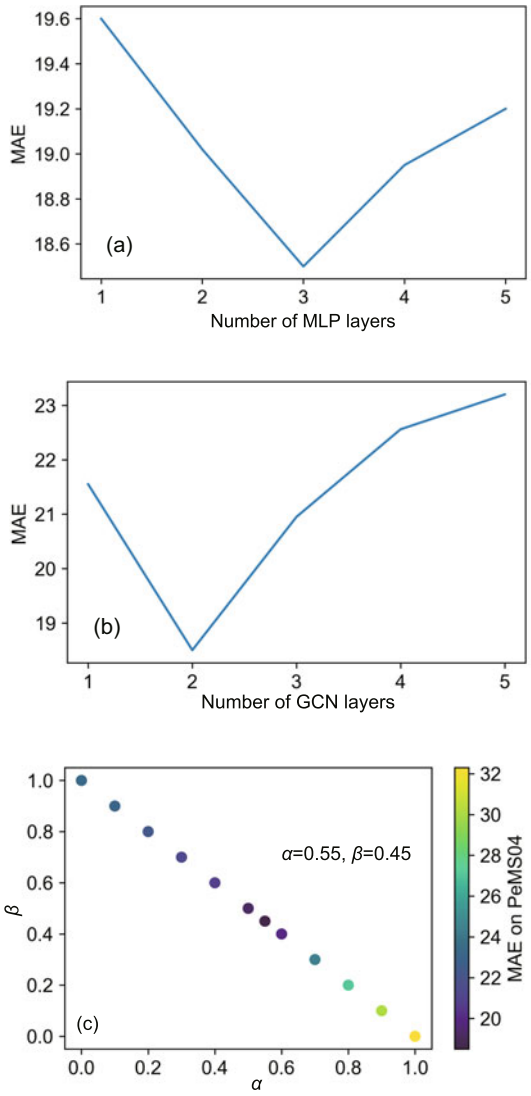

Fig. 8 Parameter adjustment process on the validation set: (a) MLP; (b) GCN; (c) weights of temporal and spatial features

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

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

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

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

where $\mathrm{Time}_{\mathrm{start}}$ stands for the real time before training and $\mathrm{Time}_{\mathrm{end}}$ represents the real time after convergence.

5.4 Ablation study of the proposed GM

In this subsection, we describe a comprehensive ablation experiment to verify the performance of GM. This does not involve the comparison of algorithm usage for OTL. The following baselines are widely used for time-series prediction or proposed in recent years, and all hyperparameters 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 YG et al., 2018).

STGCN: This baseline is a spatiotemporal GCN that uses GCNs and temporal convolution to capture spatial and temporal correlations, respectively (Yu et al., 2017).

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 spatiotemporal graphs with MLPs (Lee et al., 2024).

STAEformer: This baseline can yield outstanding results with vanilla Transformers (Liu HC et al., 2023).

The experimental results of MAE, MAPE, and RMSE over 12 predictive steps on the PeMS04 data set are shown in Table 3. We can see that deep learning methods are better than traditional model-driven methods 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 considers only temporal or spatial characteristics is not as good as a model that considers both spatial and 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 data set PeMS04, GM has reached the current research level, which enables the proposed GM to play a role in OTL.

Table 3 Metrics of all methods on PeMS04

Method	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**

The best results are in bold

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 studies have indeed improved the accuracy, but at the expense of higher algorithm complexity (as discussed in Section 2.2) and longer algorithm convergence time. GCN, VAR, and MLP also lead to little convergence time and small parameter numbers, but they fail to reach the state-of-the-art accuracy as mentioned above. However, our proposed GM considers the spatiotemporal correlation while simplifying the models and reducing the number of layers to achieve a trade-off between accuracy and time complexity. As shown in Figs. 8a and 8b, 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 conflicts with our goal of a lightweight model.

Table 4 Time and number of parameters of all methods on PeMS04

Method	Time (s)	Number of 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

5.5 Stability of the proposed GM

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

To further evaluate the training dynamics of the model, Fig. 10 shows the training and validation loss across different numbers of 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 data sets at each epoch. The loss function used was normalized MSE. The model converges quickly and the loss of the validation set is slightly higher than that of the training set.

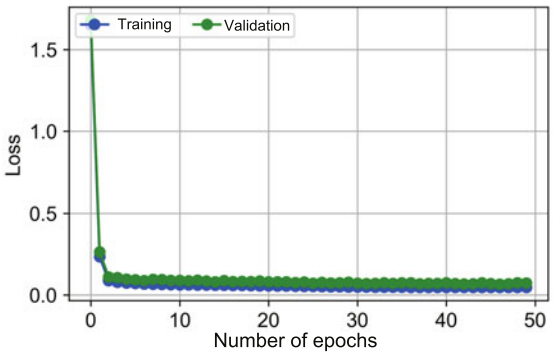

Fig. 10 Loss during training and validation across different numbers of epochs (References to color refer to the online version of this figure)

5.6 Ablation study of the proposed transfer learning

For ablation experiments on transfer learning, we compared the convergence time of algorithms us-

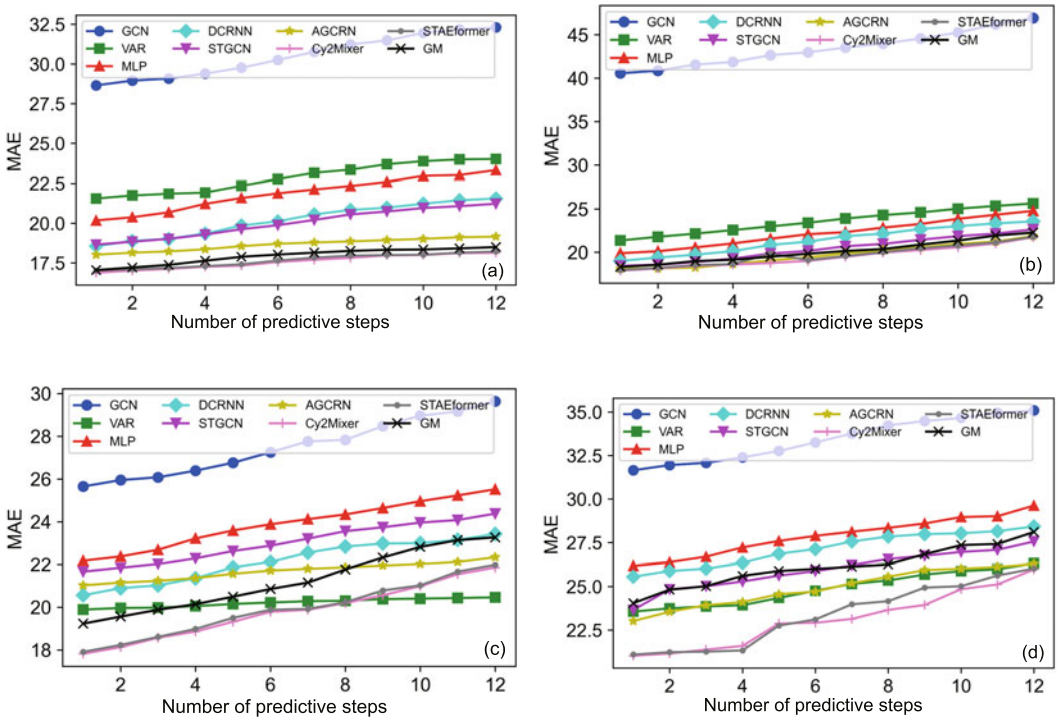

Fig. 9 Accuracy comparison as the prediction horizon varies: (a) PeMS04 (whole); (b) PeMS04 (weekday); (c) PeMS04 (holiday); (d) PeMS04 (weather)

ing 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

Degree	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	–
60%	8.64	–
70%	11.56	–

TL: transfer learning

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 data sparsity reaches 50%, the algorithm without transfer learning no longer converges, which results in traffic flow prediction 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 data collected in real time to avoid 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 GCN is $O(N)$, where N represents the number of nodes in the actual transportation network, and the time complexity of each layer of GCN is $O(N)$, so the time complexity is still $O(N)$. 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 and space complexity are both $O(N^2)$. Because MLP is a time model, the time and space complexities are related to the input data scale, so in this study the time and space complexities are $O(1)$. DCRNN consists of diffusion convolution and an RNN. After ignoring constants and lower-order variables, the space complexity of DCRNN is $O(N^2)$ and the time complexity is $O(N)$. STGCN consists of a GCN and GRU; the time complexity is $O(N)$ and the space complexity is $O(N^2)$. 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, spatiotemporal self-attention, Transformer, and position coding. The space and time complexities are both $O(N^2)$. Our proposed GM uses a sparse matrix to store traffic nodes, so the space complexity is $O(N)$. MLP has been discussed above, and the time complexity is $O(1)$.

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

5.8 Comparison with transfer learning approaches

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

ST-GFSL: This technique generates non-shared parameters based on node-level meta knowledge (Lu et al., 2022).

ST-DAAN: This technique maps the raw spatiotemporal data of the source domain and then employs domain adaptation (Wang et al., 2022).

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

Table 6 Performance and inference time comparison

Method	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

The results demonstrate that while transfer learning methods like those proposed by Lu et al.

(2022) and Wang et al. (2022) achieve high prediction accuracy, they require substantial computational resources. Lu et al. (2022) involved weighted matrix calculation parameters and reconstructed graphs, and Wang et al. (2022) involved 3D convolution. Our approach, on the other hand, achieves a balance between accuracy and efficiency by calculating the 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 issue of data sparsity has been alleviated to a certain extent. However, if the algorithm uses only 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 the prediction accuracy.

Here the data sparsity is set to 30%. We still use GM to conduct experiments on the PeMS04 (weekday) data set. The results are shown in Fig. 11.

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

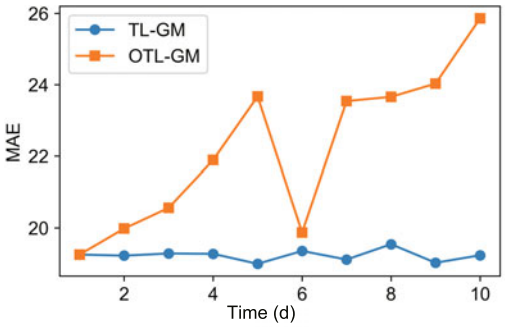

Fig. 11 Comparison between transfer learning and OTL

5.10 Results of methods with OTL

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

PeMS04 (weekday): As shown in Fig. 12, the accuracy metrics of each method have been improved to a certain extent. For MAE, the accuracy has increased from 0.48% (OTL-STGCN) to 31.54% (OTL-GCN); for MAPE, the accuracy has increased from 7.03% (OTL-STGCN) to 28.51% (OTL-GCN); for RMSE, the accuracy has increased from 1.16% (OTL-MLP) to 27.50% (OTL-HA). The proposed method OTL-GM has an improvement of 12.74%, 10.20%, and 13.56%, respectively.

PeMS04 (holiday): As shown in Fig. 13, the results are similar to those in Fig. 12. For MAE, the accuracy has increased from 1.13% (OTL-MLP) to 18.26% (OTL-GM); for MAPE, the accuracy has increased from 0.21% (OTL-HA) to 20.19% (OTL-GM); for RMSE, the accuracy has increased from 1.32% (OTL-GCN) to 15.10% (OTL-Cy2Mixer). The proposed method OTL-GM has an improvement of 18.26%, 20.19%, and 14.78%, respectively.

PeMS04 (weather): As shown in Fig. 14, the results are similar to those in Figs. 12 and 13. For MAE, the accuracy has increased from 5.16% (OTL-GCN) to 26.18% (OTL-GM); for MAPE, the accuracy has increased from 0.86% (OTL-DCRNN) to 29.72% (OTL-STAEformer); for RMSE, the accuracy has increased from 3.00% (OTL-Cy2Mixer) to 27.10% (OTL-GM). The proposed method OTL-GM has an improvement of 26.18%, 27.38%, and 27.10%, respectively.

Compared to the entire PeMS04 data set, the sample size of three classified data sets decreases

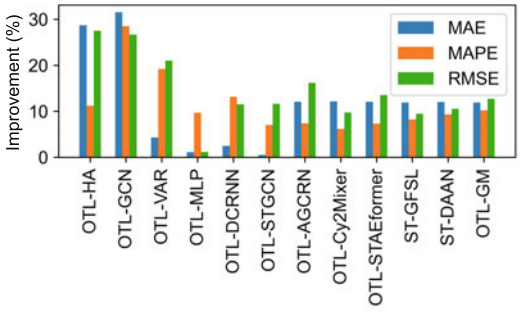

Fig. 12 Improvement of metrics on the PeMS04 (weekday) data set

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

Method	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**

The best results are in bold

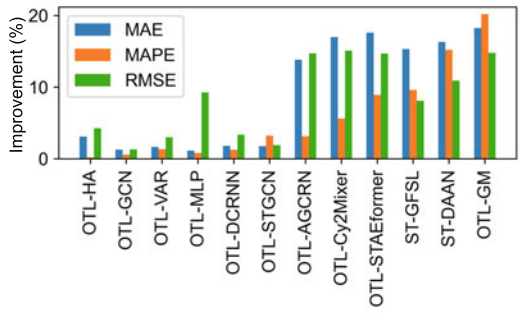

Fig. 13 Improvement of metrics on PeMS04 (holiday)

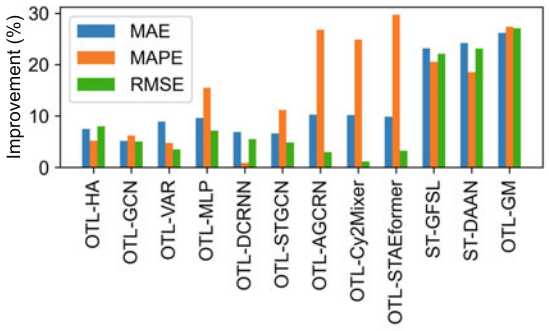

Fig. 14 Improvement of metrics on PeMS04 (weather)

(43, 16, and 6 d, respectively), and the accuracy of all algorithms decreases, which is in line with the rules of deep learning. Based on the above results, we can conclude that, compared to merging heterogeneous data sets with chaotic distributions, using OTL algorithms to merge data sets with similar data distributions has led to a certain degree of improvement in prediction accuracy, which makes the prediction results valid. Note 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 been reduced considerably. Fig. 16 presents the convergence time of all algorithms. The analysis of all the data sets is as follows:

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

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

PeMS04 (weather): The reduction in the convergence time of all algorithms has experienced a substantial improvement, from 24.77% (OTL-VAR) to 94.32% (OTL-GM).

Through an analysis combining prediction 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 the convergence time, which can be attributed to a pretrained

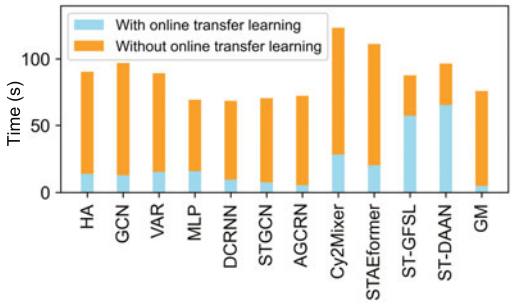

Fig. 15 Comparison of convergence time of all methods on PeMS04 (weekday)

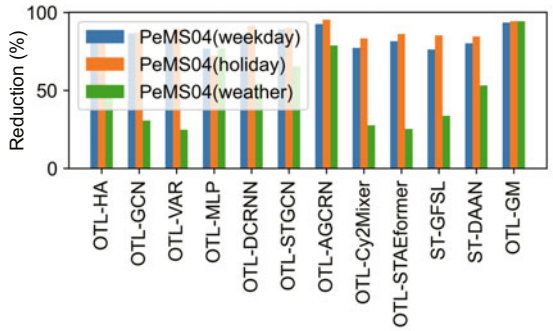

Fig. 16 Reduction in the convergence time on all the classified data sets

task algorithm.

As mentioned previously, many recent 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, practical scenarios need to take the limited device resources into account. The model and methods we propose aim to achieve state-of-the-art accuracy while, to some extent, reducing algorithm convergence time and increasing the practicality of traffic prediction.

Although we perform the pretrained tasks on our intelligent devices (as shown in Fig. 2), in the real-world scenario, pretraining on the edge servers with rich computing resources is a good choice. The OTL with a smaller Euclidean distance addresses, to some extent, the issue of data sparsity resulting from extreme weather conditions, and the shortened algorithm convergence time 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 meaningful. The OTL method we propose can be applied not only to GM, but to future advanced models proposed thereafter.

5.11 Analysis of sensitivity to MLP width

As mentioned in Section 5.2, the optimal depths for GCN and MLP are 2 and 3 respectively, and the width of GCN is the number of traffic nodes. This subsection will discuss the sensitivity to MLP width.

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

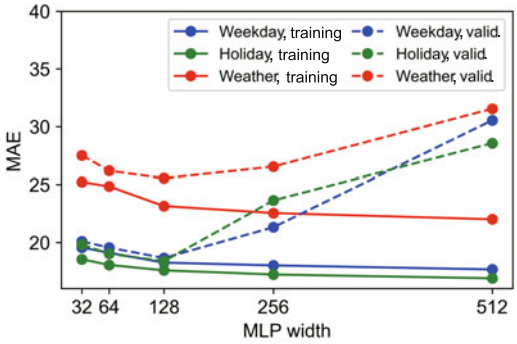

Fig. 17 Variation of MAE with different MLP widths

5.12 Analysis of scalability and performance of GM under operational conditions

In real applications, scalability is a crucial aspect of deploying the OTL-GM framework. As traffic and the number of 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, 2.20, and 4.93 s, respectively. The amount of data in PeMS04 (weekday) is 7 times that of PeMS04 (weather), but the time has increased by

only about 3 times. So, GM is well suited for a transportation network with an increasing data volume. As shown in Fig. 9, when the step size increases, the accuracy gradually decreases, still better than that of most methods.

Addressing data sparsity is one of the goals of this study. We explore 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%, GM no longer converges. However, transfer learning can successfully compensate for the sparsity 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. Using the Euclidean distance, we verified the similarity in the 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 data set confirmed 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 showed that the proposed OTL approach still worked well. Despite this, ablation studies indicated that the proposed OTL approach remained effective. If a diverse set of data samples is available, it has the potential to improve prediction accuracy and reduce algorithm convergence time. We performed experiments only on our intelligent device and used OTL in real-world scenarios. Pretraining on edge servers with rich computing resources is a wise choice when combined with IoT technology. In the future, it is meaningful 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. Chendingying LU drafted the paper. Yichuang SUN helped organize the paper. Chendingying LU revised and finalized the 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.

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 Annual Conf on 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 on 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 on 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, 2016. 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. Proc 33rd 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 Int 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. *Cog 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, 2021. 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.356-360.

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 Y, Long PM, 1999. The relaxed online maximum margin algorithm. Proc 13th Int Conf on Neural Information Processing Systems, p.361-387.

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.

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, 2004. 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 Tut*, 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 Int Things J*, 9(24):25685-25697. https://doi.org/10.1109/JIOT.2022.3196461

Sutskever I, Vinyals O, Le QV, 2014. 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 Iss*, 22(E-C168):22-41.

Vaswani A, Shazeer N, Parmar N, et al., 2017. 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*, 24(8):8592-8605. https://doi.org/10.1109/TITS.2023.3250424

Yu B, Yin HT, Zhu ZX, 2017. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. Proc 27th Int Joint Conf on Artificial Intelligence, p.3634-3640.

Zhang CY, Patras P, 2018. 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

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. In: Modeling Financial Time Series with S-PLUS®. Springer, New York, p.385-429. https://doi.org/10.1007/978-0-387-32348-0_11