electronics

MDPI

Article

Multi-Spatio-Temporal Convolutional Neural Network for Short-Term Metro Passenger Flow Prediction

Ye Lu [1], Changjiang Zheng [1,*], Shukang Zheng [2], Junze Ma [1], Zhilong Wu [1], Fei Wu [3] and Yang Shen [3]

[1] College of Civil and Transportation Engineering, Hohai University, Xikang Road, Nanjing 210024, China; 211304050005@hhu.edu.cn (Y.L.)
[2] College of Environment, Hohai University, Xikang Road, Nanjing 210024, China
[3] College of Computer and Information, Hohai University, 8 Focheng West Road, Nanjing 211100, China; 210207090003@hhu.edu.cn (F.W.)
* Correspondence: zheng@hhu.edu.cn; Tel.: +86-139-0516-7096

Abstract: Accurate short-term prediction of metro passenger flow can offer significant assistance in optimizing train schedules, reducing congestion during peak times, and improving the service level of the metro system. Currently, most models do not fully utilize the high-resolution data aggregated by automatic fare collection systems. Therefore, we propose a model, called MST-GRT, that aggregates multi-time granularity data and considers multi-graph structures. Firstly, we analyze the correlation between metro passenger flow sequences at different time granularities and establish a principle for extracting the spatiotemporal correlation of data at different time granularities using the multi-graph neural network. Subsequently, we use residual blocks to construct a deep convolutional neural network to aggregate the data of different time granularities from small to large, obtaining multi-channel feature maps of multi-time granularity. To process the multi-channel feature maps, we use 2D dilated causal convolution to reconstruct the TCN (Temporal Convolutional Network) to compress the channel number of the feature maps and extract the time dependency of the data, and finally output the results through a fully connected layer. The experimental results demonstrate that our model outperforms the baseline models on the Hangzhou Metro smart-card data set.

Keywords: multi-time granularity; graph neural network; residual block; temporal convolutional network; passenger flow

Citation: Lu, Y.; Zheng, C.; Zheng, S.; Ma, J.; Wu, Z.; Wu, F.; Shen, Y. Multi-Spatio-Temporal Convolutional Neural Network for Short-Term Metro Passenger Flow Prediction. *Electronics* **2024**, *13*, 181. https://doi.org/10.3390/electronics13010181

Academic Editor: Ping-Feng Pai

Received: 29 November 2023
Revised: 21 December 2023
Accepted: 28 December 2023
Published: 30 December 2023

1. Introduction

With the continuous acceleration of urbanization, the construction of urban rail transit systems with large capacity, medium-to-long distance, high speed, punctuality, and high passenger comfort has become the main solution to city traffic congestion and a major driver of urban development. The metro, as the main form of urban rail transit, meets the travel needs of most city residents. According to statistics, the metro passenger flow in Hangzhou, China, reached 622.9 million person-times in the first half of 2023, a 45.46% increase from 2022 and a 112.24% increase from 2019. The dramatic increase in passenger flow brings more challenges to the operation and management of the metro, making metro passenger flow prediction increasingly important. For metro management, accurate passenger flow prediction can serve as an important basis for planning train schedules, station services, and personnel dispatch, thus easing station congestion during peak passenger flow periods through reasonable operational planning. For passengers, accurate passenger flow prediction can help them plan their travel time, avoid peak subway periods, and reduce waiting time, thereby improving subway riding satisfaction and increasing passenger loyalty to the subway system. Additionally, Guangzhou, China, initiated a pilot for night metro logistics in July 2022. Accurate passenger flow prediction can be used to calculate the remaining capacity of subway trains, creating the possibility for passenger and cargo

co-transportation during non-peak subway periods, thereby improving urban logistics efficiency while meeting passenger travel demands.

Metro passenger flow datasets contain rich spatio-temporal features, and its variation is influenced by the metro network topology and other factors such as weather. Traditional statistical-based prediction methods like the Autoregressive Integrated Moving Average Model (ARIMA) [1] and its variants, like Seasonal Autoregressive Integrated Moving Average (SARIMA) [2], have fixed model complexity and fewer parameters, making it difficult to capture non-linear features in the data. They are also limited by the assumption of data stability and cannot consider the spatio-temporal correlation of the data, making it challenging for these traditional statistical prediction models to meet the performance demands of short-term metro passenger flow prediction tasks. The advent of machine learning techniques has brought more flexible, accurate, and automated modeling methods to time series prediction, capable of modeling more complex data and better handling the randomness and non-linearity of passenger flow changes. For example, the K-Nearest Neighbor (KNN) [3] model is a simple, intuitive, and versatile classification and regression algorithm based on the nearest neighbor relationship and is suitable for various data types; Support Vector Machine (SVM) [4] is a model that can adapt to different data distributions by selecting different kernel functions; and the Neural Network (NN) [5] model can learn complex mapping relationships from input to output through weight training. With the continuous development of neural networks, deep learning neural networks that have more hidden layers and can learn higher-level data features and more complex data relationships have become mainstream.

Recurrent Neural Networks (RNNs) are commonly used to model temporal dependencies in data. However, the inherent design flaws of standard RNN can lead to issues such as vanishing and exploding gradients when handling long sequences, making model training difficult and limiting its application in many tasks [6,7]. To address these issues, Long Short-Term Memory (LSTM) [8] networks with gating mechanisms were introduced and have demonstrated good performance in tasks such as traffic flow prediction. The Gated Recurrent Unit (GRU) [9], by simplifying the gate structure of the LSTM and reducing network parameters, has effectively improved training efficiency. Experimental evidence shows that the GRU model outperforms the LSTM model in some sequence modeling tasks. In addition, Convolutional Neural Network (CNN) based neural networks, such as the Temporal Convolutional Network (TCN) [10], which uses dilated convolutions in combination with causal convolutions, and Gated Linear Units (GLUs) [11], which use CNN and gating mechanisms for parallel processing of time series data, can effectively extract temporal features from data. However, the above models only consider data in terms of temporal characteristics, ignoring data with an implicit spatial dependency, such as metro passenger flow data. To capture spatial features of the data, CNNs must be used. CNN-based networks have translational invariance and are only able to process data in Euclidean form. Therefore, when modeling traffic flow data, the city must be viewed as a graph with many grids, and the traffic flow is considered as a feature of each grid to extract the spatial features of the data [12,13]. In order to apply the concept of CNNs to graph-structured data, Graph Neural Networks (GNNs) were proposed and have shown superior performance in traffic flow prediction tasks. The Chebyshev Graph Convolutional Network (ChebNet) [14] uses Chebyshev polynomials to replace the spectral convolution kernel for Chebyshev convolution operations; however, as the order of Chebyshev polynomials rises, the computational cost of ChebNet increases quickly, making it difficult to apply to large graph-structured data. Graph Convolutional Networks (GCNs) [15] are first-order approximations of ChebNet based on spectral convolutions on graphs; they use the Chebyshev polynomial of the eigenvalue diagonal matrix to approximate filters, significantly reducing the cost of performing graph convolution operations on large graph-structured data. The Graph Attention Network (GAT) [16] allocates weights to neighboring nodes dynamically using attention mechanisms, enabling the model to focus on more important neighboring

nodes. Graph SAmple and aggreGatE (GraphSAGE) [17] transforms transductive learning into inductive learning by learning how nodes aggregate from their neighborhoods.

In order to capture both spatial and temporal features of data simultaneously, various neural network models combining CNNs and RNNs have been proposed and have shown excellent performance [18]. The Temporal Graph Convolutional Network (T-GCN) [19] combines GCN and GRU to extract the spatial-temporal features of urban road network traffic flow. The Diffusion Convolutional Recurrent Neural Network (DCRNN) [20] captures spatial features of traffic flow through bidirectional random walks on the graph and captures its temporal features using the encoder-decoder architecture with scheduled sampling. The Spatiotemporal Attention-Based Graph Convolution Network (AST-GAT) [21] uses GAT to capture spatial dependencies between road sections, while using LSTM with attention mechanisms to capture time dependencies, achieving traffic speed prediction based on road sections. Furthermore, CNN-based convolutional neural network models can also capture both spatial and temporal features of data simultaneously. For example, Spatio-Temporal Graph Convolutional Networks (STGCNs) [22] are composed of two spatial convolution blocks using GCN and one temporal convolution block using causal convolution and GLU; Multivariate Time Series Graph Neural Networks (MTGNNs) [23] capture spatial and temporal dependencies through a mix-hop propagation layer and dilated inception layer; and the Multi-View Multi-Attention Graph Neural Network (AMGC-AT) [24] uses multi-graph GCN with two different self-attention mechanisms and GLU for subway passenger flow prediction.

However, existing models have not fully utilized other time granularity data apart from the time granularity of the prediction target. Traditional GNN models usually focus on the local neighborhood relationship of nodes, ignoring the global features of the entire graph. RNN-based models for modeling time dependencies have issues with gradient vanishing or exploding when dealing with long-term dependencies and are prone to overfitting. To overcome these drawbacks, we propose a deep learning architecture called MST-GRT, which consists of multi-graph convolution modules, multi-time scale data aggregation modules, and TCN modules. In the multi-graph convolution modules, spatial features of the data are fully extracted using three different perspectives of graph structures. In the multi-time scale data aggregation modules, a deep convolutional network is constructed using residual blocks to aggregate data from small to big time granularities, generating multi-channel feature maps that contain features with multi-time granularity. In the TCN module, 2D dilated causal convolutional layers are used to reconstruct TCN, which extracts temporal features of the data while considering the relationships among different channel feature maps, thus enhancing the model's prediction performance while maintaining the shape of the feature map. The model is evaluated using the Hangzhou Metro smart-card dataset, and the experimental results show that the proposed model outperforms other baseline models in short-term passenger flow prediction tasks in the metro system, particularly at 60 min time granularity. The main contributions of this research are as follows:

1. In this paper, the innovative idea of utilizing multi-time granularity data is proposed, pointing out the challenges in aggregating multi-time granularity data and providing a feasible method for aggregating such data.
2. A deep convolutional network is constructed using residual blocks, and the width of different time granularity feature maps is controlled by adjusting the stride parameter in the convolutional layers of the residual blocks to aggregate multi-time granularity data. The ablation experiment results demonstrate that this network outperforms the use of fully connected-layer or single-layer convolutional networks in aggregating multi-time granularity data.
3. During the construction of the deep convolutional neural network, a balance is found between model prediction accuracy and computational resource consumption.

2. Preliminaries and Data

2.1. Problem Formulation

We refer to the time interval at which metro passenger flow data is recorded (in minutes) as the time granularity. In multi-time granularity data, data with smaller time granularity can provide supplementary details for data with larger time granularity [25]. Fully utilizing multi-time granularity data plays a crucial role in improving the accuracy of short-term passenger flow prediction models. Figure 1 depicts the passenger inflow at four different time granularities. From the figure, it can be seen that data with larger time granularity reflects the overall trend of the passenger inflow, while data with smaller time granularity reflects the fluctuations of the passenger inflow.

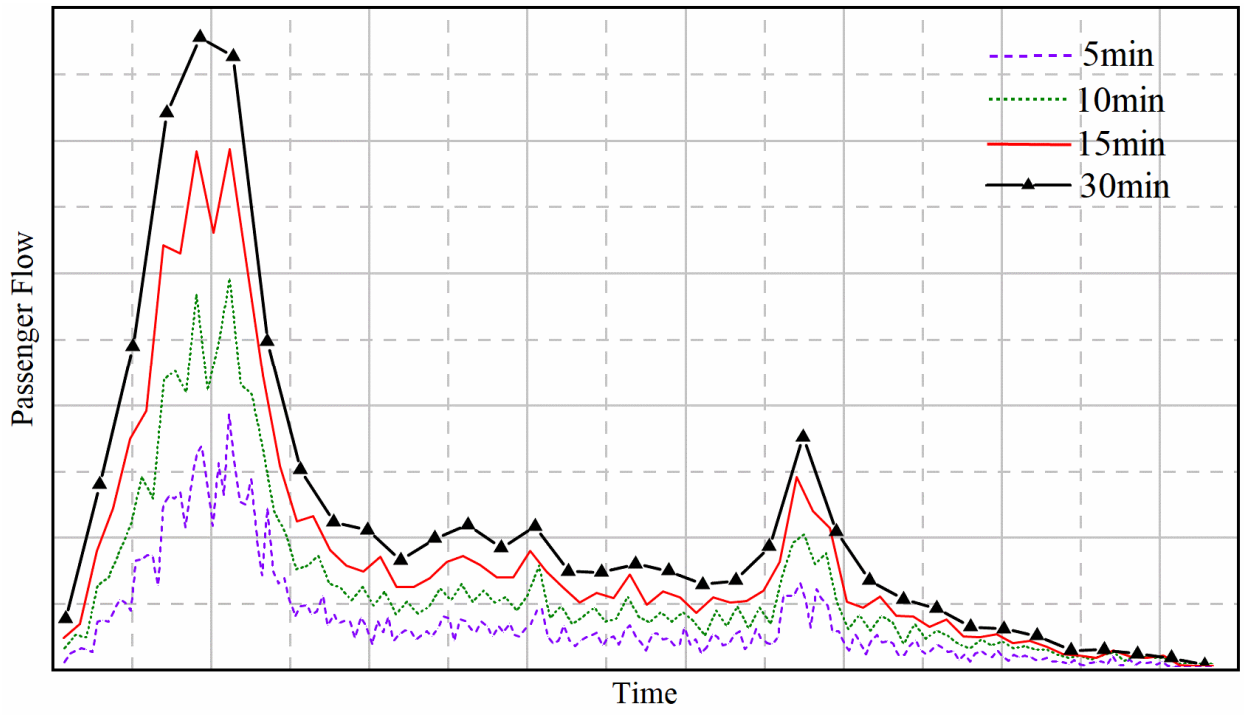


Figure 1. Multi-time granularity passenger flow data.

We refer to the time granularity corresponding to the prediction target as the target granularity, and all other time granularities with granularity smaller than the target granularity are referred to as auxiliary granularities. We aggregate data from all auxiliary granularities with data from the target granularity to construct the multi-time granularity passenger flow feature maps. However, not all subsidiary data can be perfectly aggregated with other time granularity data. Figure 2 depicts five different time granularities, where 10 min is the target granularity, and 1 min is the smallest auxiliary granularity. Therefore, the 10 min target granularity should include 1 min–9 min, for a total of 9 auxiliary granularities. Figure 2 visualizes the relationship of 4 auxiliary granularities with the target granularity. To match the shape of the target granularity data, the 6 min auxiliary granularity data need to trim the excess part, and the 3 min auxiliary granularity data need to make up for the missing part. To avoid the disturbance that may be caused by the trimmed or filled part in the auxiliary granularity data, and considering that too many auxiliary granularities could make the model too deep and increase the training cost, we choose the factors of the target granularity as auxiliary granularities (e.g., the auxiliary granularities of 15 min target granularity are 1 min, 3 min, and 5 min). The multi-time granularity passenger flow sequence set is defined as $\{a_1, a_2, \ldots, a_{k-1}, a_k\}$, where a_k represents the target granularity passenger flow sequence, and a_1 to a_{k-1} is the auxiliary granularity passenger

flow sequence, satisfying $a_1|a_k, a_2|a_k, \ldots, a_{k-1}|a_k$. The term $A|B$ means B is divisible by A (i.e., the target granularity can be aggregated from the auxiliary granularity).

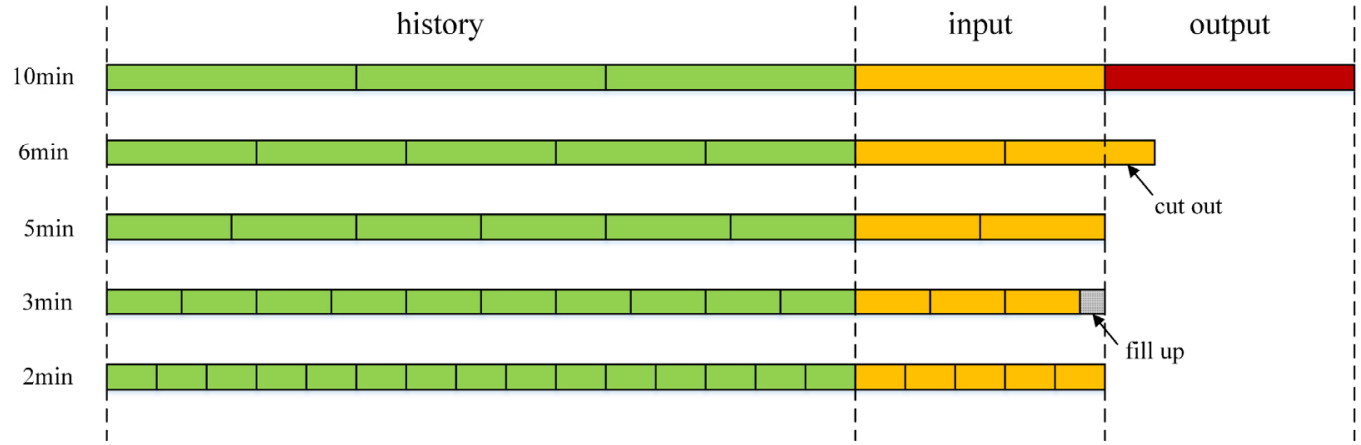


Figure 2. Characteristics of passenger flow data aggregation with different time granularities.

We define the metro network as graph $G = (V, E, A)$, where V represents the set of metro stations, $V = \{v_1, v_2, v_3, \ldots, v_n\}$, where n represents the station number, E represents the set of edges between metro stations, and $A \in \mathbb{R}^{n \times n}$ is a symmetric matrix containing only 0 and 1 elements, representing the connection relationship between metro stations. At time t, the passenger flow status at all stations in G can be represented as a collection of multi-time granularity feature maps F_t, $F_t = \left\{X_t^1, X_t^2, \ldots, X_t^k, \ldots X_t^K\right\}$, where X_t^K represents the passenger flow feature map of all stations at time t under the target granularity K, $X_t^K \in \mathbb{R}^{n \times m}$, where n represents the height of the feature map, which corresponds to the number of nodes, and m represents the width of the feature map, which corresponds to the feature dimension of each node. X_t^1 is the smallest auxiliary granularity feature map at time t, $X_t^1 \in \mathbb{R}^{n \times (m \cdot K)}$, the width of X_t^1 is $m \cdot K$; X_t^k is the feature map of other auxiliary granularities at time t, $X_t^k \in \mathbb{R}^{n \times (m \cdot K/k)}$, the width of X_t^k is $m \cdot K/k$. Therefore, the short-term metro passenger flow prediction problem can be described as learning the function f, which maps the multi-time granularity passenger flow feature maps of T historical time points to the target granularity passenger flow feature map of one future time point. The following formula can be used for description:

$$X_{t+1}^K = f([F_{t-T+1}, F_{t-T+2}, \ldots, F_t]; G) \tag{1}$$

2.2. Data

We validate the effectiveness of our MST-GRT model using the Hangzhou Metro smart-card dataset. This dataset includes approximately 70 million swipe records from 80 effective stations on three metro lines over 26 days, from 1 January to 26 January 2019. The distribution of metro lines and stations are shown in Figure 3. Based on experience, effectively controlling the inflow of passenger flow can help control the outflow of passenger flow. Therefore, this paper focuses on studying the inflow of passenger flow. We first gather the swipe data in 1 min time granularity and then aggregate other time granularity data based on the 1 min time granularity data. We conducted a simple analysis of the data at the 10 min time granularity and found that only 0.91% of the passenger flow data from 6:00 to 23:30 is zero. Therefore, we selected the data from 6:00 to 23:30 each day as our final dataset.

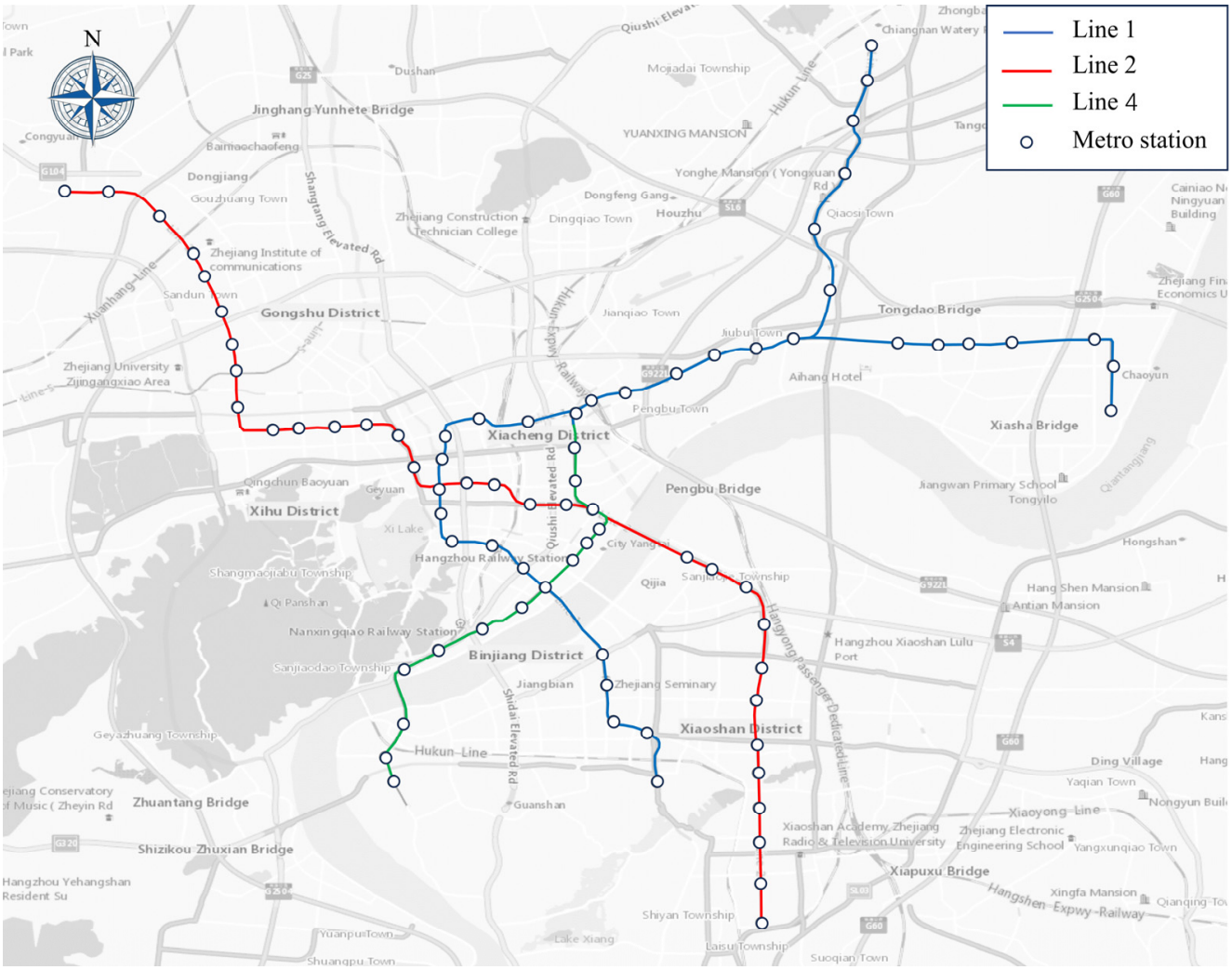


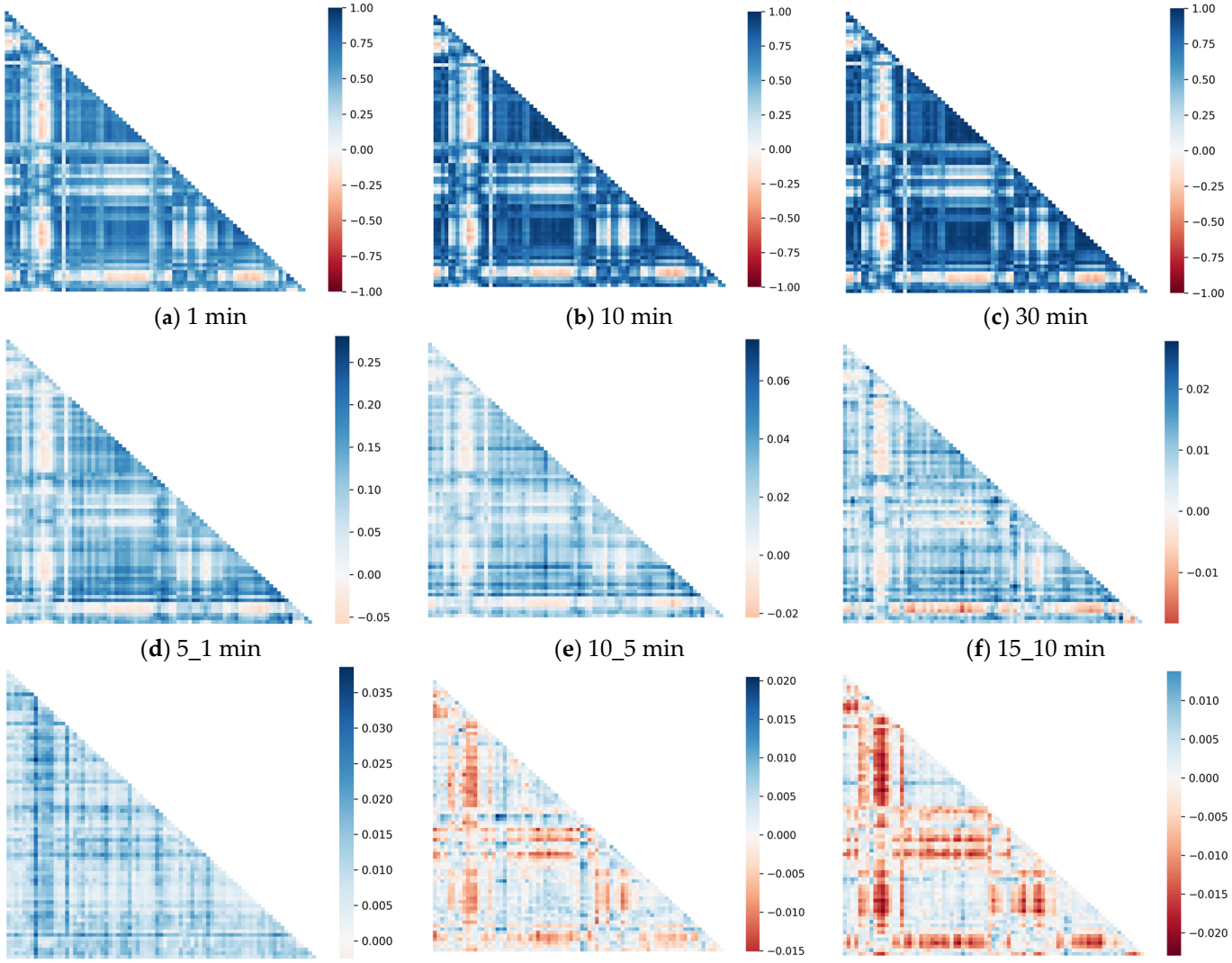
Figure 3. Distribution of metro lines and stations.

Due to the use of multi-time granularity data in our MST-GRT model, including data with small time granularity, the width of the feature map tends to be large when the time granularity is small. If we use multi-graph GCN to extract spatial features from all the time granularity data, it could lead to an excessive number of parameters in the multi-graph convolutional layers and the aggregation of overly redundant node features. Therefore, we analyze the correlations of passenger flow sequences at different time granularities to determine the appropriate size of time granularity to use multi-graph GCN for extracting spatial features. For data that are continuous in time, we can calculate the correlation between data using the Pearson Correlation Coefficient (PCC):

$$A_{PCC,ij} = \frac{\sum_{k=1}^{n}\left(X_{i,k}-\overline{X_i}\right)\left(X_{j,k}-\overline{X_j}\right)}{\sqrt{\sum_{k=1}^{n}\left(X_{i,k}-\overline{X_i}\right)^2}\sqrt{\sum_{i=1}^{n}\left(X_{j,k}-\overline{X_j}\right)^2}} \tag{2}$$

where $A_{PCC,ij}$ represents the PCC between the passenger flow sequences of station i and station j, and $\overline{X_i}$ and $\overline{X_j}$ represent the averages of the passenger flow sequences of station i and station j.

The Hangzhou Metro smart-card dataset contains data from 80 stations. From the PCC calculation formula, we can conclude that A_{PCC} is an 80 × 80 symmetrical matrix with the elements on the main diagonal equal to 1. To better compare the A_{PCC} at different time granularities, we mask the elements in the upper triangle and on the main diagonal of A_{PCC} and plot the heatmap as shown in Figure 4. Figure 4a–c intuitively shows that the correlation of passenger flow sequences at 10 min and 30 min granularity is greater than that at 1 min granularity, which aligns with experience. Figure 4d shows a significant increase in the correlation of passenger flow sequences from 1 min to 5 min granularity. Figure 4e–g shows that from 5 min to 20 min granularity, there is a slight increase in the correlation of passenger flow sequences at most stations. However, Figure 4h,i shows a slight decline in the correlation of passenger flow sequences at most stations from 20 min to 30 min granularity. From this, we can conclude that from 1 min to 5 min granularity, the correlation of passenger flow sequences at each time granularity rises rapidly; from 5 min to 20 min granularity, the growth of correlation slows down and gradually stabilizes; from 20 min to 30 min granularity, the correlation of passenger flow sequences at most stations begins to decline slightly, but it is still higher compared to the correlation of passenger flow sequences between 1 min and 5 min granularity.



(**a**) 1 min (**b**) 10 min (**c**) 30 min

(**d**) 5_1 min (**e**) 10_5 min (**f**) 15_10 min

(**g**) 20_15 min (**h**) 25_20 min (**i**) 30_25 min

Figure 4. (**a**–**c**) Respectively represent the A_{PCC} at time granularities of 1 min, 10 min, and 30 min. (**d**) Represents the difference between A_{PCC} at 5 min time granularity and A_{PCC} at 1 min time granularity; the meanings of (**e**–**i**) can be inferred accordingly.

3. Methods

3.1. Overall Architecture

The overall architecture of the MST-GRT model is shown in Figure 5. The model framework consists of three main parts: the GCN Block for extracting spatial features, the FE Block for extracting features and aggregating data at different time granularities, and the TCN Block for extracting temporal features. We concatenate multi-granularity data from small to large time granularity in channels. In the aggregated multi-channel feature maps, the smaller the time granularity, the higher the feature level. According to the analysis in Section 2.2, the smaller the time granularity, the lower the spatio-temporal correlation of the data; by weighing the computational cost and the prediction effect, we decided to disregard the spatial features of auxiliary granularities with the time granularity smaller than 5 min. For these data, we use the Conv Block, which processes them using three different sizes of convolutional kernels to extract features of different scales and concatenates them along the channel dimension. For auxiliary granularities with time granularity greater than or equal to 5 min and the target granularity data, we use the GCN Block to extract spatial correlations. This module considers three aspects: node position, network topology structure information, and similarity of passenger flow. We concatenate the original data with the data processed by multi-graph GCN along the channel dimension and expand the number of channels through a 2D convolutional layer with a kernel size of 1. The FE Block utilizes the residual block structure to prevent the potential issues of gradient vanishing and exploding during the training of deep networks. By adjusting the stride parameter of the convolution operation in the residual blocks, we can control the width of the feature map after convolution to match the width of the feature maps of other time granularities, allowing them to be concatenated along the channel dimension. However, there are cases where some auxiliary granularity data cannot be directly aggregated through convolution operations (e.g., for auxiliary granularities of 2 min and 5 min, assuming the target granularity is K, and the width of the feature map at target granularity is m, then the widths of the feature maps at these two auxiliary granularities are $m \cdot K/2$ and $m \cdot K/5$). We transform the auxiliary granularity data that cannot be directly aggregated with the next-level auxiliary granularity data through a fully connected layer (e.g., by transforming the width of the feature map at the auxiliary granularity of 2 min to $2(m \cdot K/5)$, the stride parameter of the convolution operation in the residual block can be set to (1, 2), thus matching the data at the auxiliary granularity of 5 min). To maintain the shape of the feature maps and force the model to learn a unified rule for extracting passenger flow characteristics from all metro stations in the entire metro network, we use the 2D dilated causal convolution to reconstruct the TCN to extract temporal features, compress the number of feature map channels, and finally transform the output data dimensions. The final result is obtained through a fully connected layer. To avoid high computational costs caused by too many channels in the feature maps after aggregating multiple auxiliary granularity data, we use a 2D convolution layer to compress the number of channels after each FE Block.

3.2. Spatial Dependency Modeling

To comprehensively extract deep spatial features from metro passenger flow data, we extended the traditional GCN that solely uses the neighborhood graph to a GCN utilizing multiple types of graphs. It consists of two parts: multi-graph generation and multi-graph convolution.

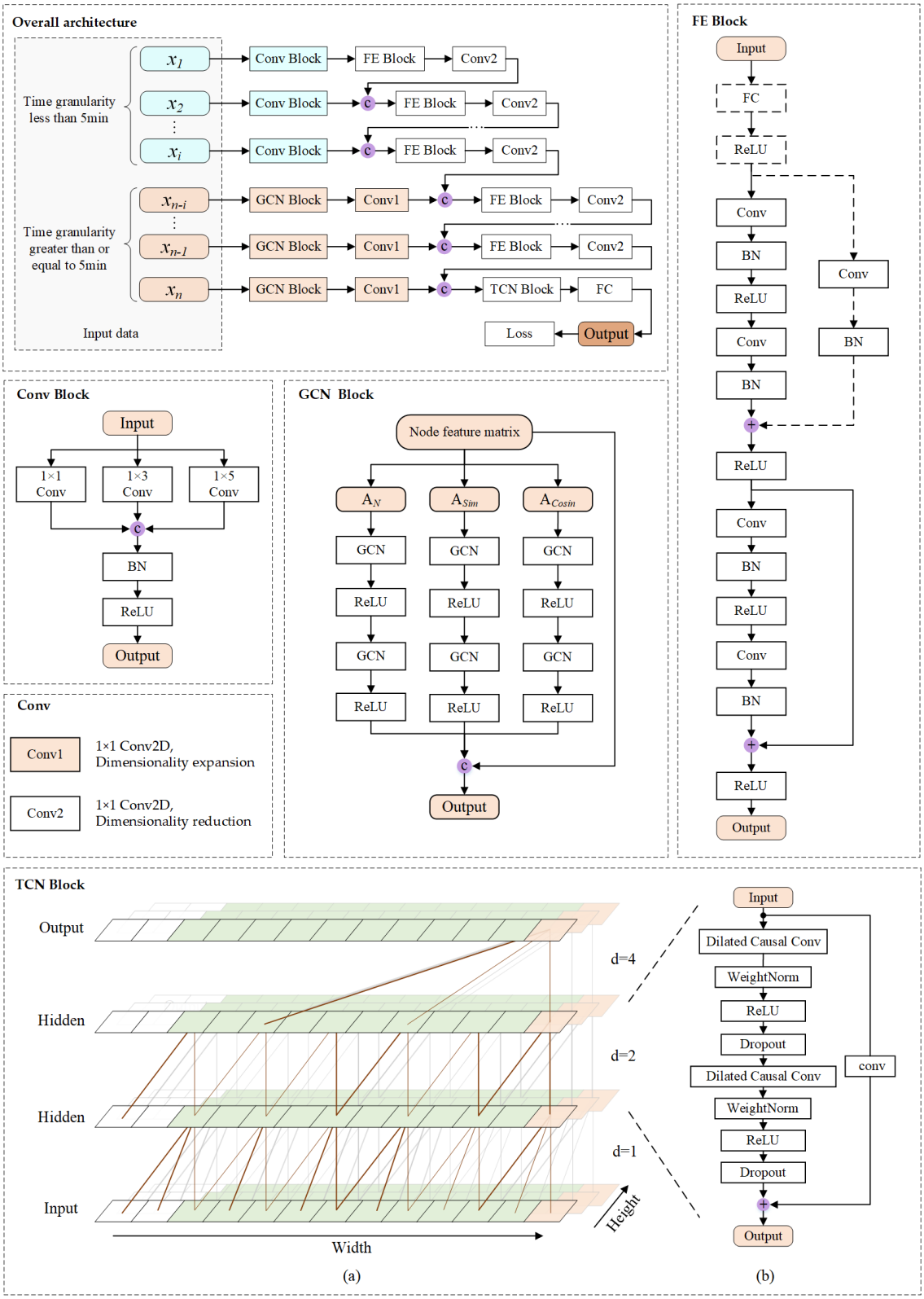


Figure 5. Overall architecture of the model. FC represents the fully connected layer, ReLU represents the ReLU activation function, Conv represents the 2D dimensional convolution, and BN represents the batch normalization layer. A_N represents the neighborhood graph, A_{sim} represents the SimRank graph, and A_{Cosin} represents the cosine similarity graph. In the TCN block, (**a**) represents the 2D dilated causal convolution with filter size k = (1,3) and dilation factors d = (1,2,4), and (**b**) illustrates how information flows between layers.

3.2.1. Multi-Graph Generation

Neighborhood Graph: We model the metro network with the L-space method, where nodes represent metro stations, and if two stations are connected, there exists an undirected edge in the network. We believe that from the perspectives of passenger travel demand, urban planning, and transportation connectivity, there is a certain correlation between the passenger flows of adjacent metro stations. The neighborhood graph can be defined as follows:

$$A_{N,ij} = 1, \text{ when } v_i \text{ and } v_j \text{ are adjacent, otherwise } A_{N,ij} = 0 \tag{3}$$

where v_i and v_j represent station i and j.

SimRank Graph: SimRank is a model based on the topological structure information of the graph to measure the similarity between any two objects [26]. We believe that two metro stations with similar structures also exhibit some degree of similarity in their passenger flows. The SimRank similarity between them can be calculated using the following formula:

$$A_{sim,ab} = s(a,b) = \begin{cases} 1 & a = b \\ \frac{c}{|I(a)||I(b)|} \sum_{x \in I(a)} \sum_{y \in I(b)} s(x,y) & a \neq b \\ 0 & I(a) = \varnothing \text{ or } I(b) = \varnothing \end{cases} \tag{4}$$

where c is the damping factor between 0 and 1, commonly in the range of 0.6 to 0.8; in this paper, it is set to 0.8. $I(a)$ represents the number of elements in the set of all neighbor nodes of node a. Generally, an initial SimRank value is assigned to all nodes, and then the SimRank values are updated by recursively considering the similarity between neighboring nodes of a node until the SimRank values converge.

Cosin Similarity Graph: The most intuitive way to measure the similarity between the passenger flow sequences of any two stations is through cosine similarity, which is commonly used to compute similarity between two vectors. The definition of cosine similarity between any two stations is

$$A^t_{Cosin,ij} = \frac{x^t_i \cdot x^t_j}{\left|x^t_i\right|\left|x^t_j\right|} \tag{5}$$

$$A_{cs} = ave(A^{10}_{Cosin} + A^{15}_{Cosin} + A^{30}_{Cosin}) \tag{6}$$

where A^t_{Cosin} represents the Cosine Similarity Graph at the time granularity t, x^t_i and x^t_j are the passenger flow sequence vectors of stations i and j at time granularity t,$ave(\cdot)$ denotes the average function, and A_{cs} is the final constructed Cosine Similarity Graph.

3.2.2. Multi-Graph Convolution

The GCN model in this paper uses the widely adopted efficient hierarchical propagation rule based on the first-order approximation of spectral convolution on the graph. This method is based on the spectral theory of graphs, using the eigendecomposition of the graph's Laplacian matrix to perform convolution operations, effectively capturing complex relationships between nodes and their neighborhoods [15]. For the neighborhood graph, the forward propagation formula of graph convolution is

$$H^{(l+1)}_n = \sigma\left(\widetilde{D}_n{}^{-\frac{1}{2}} \widetilde{A}_n \widetilde{D}_n{}^{-\frac{1}{2}} H^{(l)}_n W^{(l)}_n\right) \tag{7}$$

where $\widetilde{A}_n = A_n + I$, A_n is the adjacency matrix, I is the identity matrix, $\widetilde{D}_{n,ii} = \sum_j \widetilde{A}_{n,ij}$ and $H^{(0)}_n = X \in \mathbb{R}^{n \times f}$, where X is the feature matrix composed of the passenger flow of all stations, n is the number of stations, and f is the dimension of the feature vector for each station. $H^{(l)}_n$ represents the output of layer l, $H^{(l)}_n \in \mathbb{R}^{n \times d}$, where d represents the hidden

feature dimension of each station in layer l. ${W_n}^l$ is the weight for layer l, and $\sigma(\cdot)$ is the activation function, commonly chosen as ReLU.

Similarly, we define formalization of the convolution of the SimRank Graph and Cosine Similarity Graph as follows:

$$H_{sr}^{(l+1)} = \sigma\left(D_{sr}{}^{-\frac{1}{2}} A_{sr} D_{sr}{}^{-\frac{1}{2}} H_{sr}^{(l)} W_{sr}^{(l)}\right) \tag{8}$$

$$H_{cs}^{(l+1)} = \sigma\left(D_{cs}{}^{-\frac{1}{2}} A_{cs} D_{cs}{}^{-\frac{1}{2}} H_{cs}^{(l)} W_{cs}^{(l)}\right) \tag{9}$$

According to the definition in Section 3.2.2, the main diagonal element of both A_{sr} and A_{cs} is 1. In the convolution process, the features of the nodes themselves can already be considered, so there is no need to further process A_{sr} and A_{cs}. For the other elements that are not on the main diagonal, which are all within the interval $[0, 1)$, the method of calculating the degree matrix of the neighborhood graph is obviously not applicable here. Therefore, we define $D_{sr,ii} = \left\lceil \sum_j A_{sr,ij} \right\rceil$ and $D_{cs,ii} = \left\lceil \sum_j A_{cs,ij} \right\rceil$, where $\lceil \cdot \rceil$ represents the ceiling function.

We concatenate the convolutional features of the three graphs with the original input X to obtain the output features of the GCN Block, denoted as $\boldsymbol{H}$:

$$\boldsymbol{H} = [X, H_n^{(l+1)}, H_{sr}^{(l+1)}, H_{cs}^{(l+1)}] \tag{10}$$

where $[\cdot]$ represents the concatenation operation.

3.3. Residual Block

Deep Neural Networks (DNNs) can fit more complex mapping relationships, and the greater the number of layers in the network, the higher the level of extracted features. However, excessively deep networks are difficult to train and may suffer from degradation issues, where the performance does not improve and may even degrade compared to shallower networks. Residual Neural Networks (Resnet) [27] effectively address issues such as gradient vanishing and information flow in deep neural networks by utilizing skip connections and residual learning mechanisms, making the training of deep neural networks more manageable. The residual block is the basic unit constituting the residual neural network, and its structure is illustrated in Figure 6. In the figure, $\vec{x}$ represents the input sequence, each of layer 1 and layer 2 includes a convolutional layer and a batch normalization layer, $\mathcal{F}(\cdot)$ denotes the residual mapping function, and $a(\cdot)$ is the activation function commonly chosen as ReLU. Additionally, because skip connections directly add the original data to the learned residual, channel matching is usually achieved through a convolutional layer with a kernel size of 1.

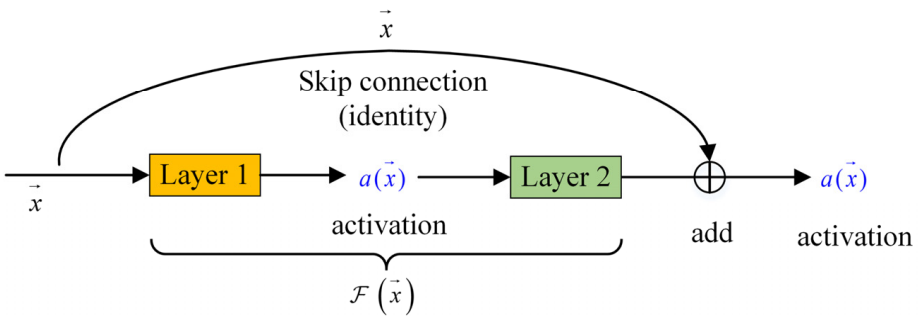


Figure 6. Residual block structure diagram.

Assuming the output of layer l is x_l, the residual block can be represented simply as

$$x_{l+1} = x_l + \mathcal{F}(x_l, W_l) \tag{11}$$

The relationship of output between any depth layer L and layer l can be expressed as

$$x_L = x_l + \sum_{i=1}^{L-1} \mathcal{F}(x_i, W_i) \tag{12}$$

Assuming the loss function is E, according to the chain rule of backpropagation, we can obtain the following:

$$\frac{\partial E}{\partial x_l} = \frac{\partial E}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial E}{\partial x_L}(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} \mathcal{F}(x_i, W_i)) \tag{13}$$

Gradient propagation can be viewed as two parts. The first part is $\frac{\partial E}{\partial x_L}$, which is independent of the parameter W, ensuring that the error signal can be propagated directly to the lower layers without any intermediate weight matrix transformation, which to some extent alleviates the problem of gradient dissemination. The second part is $1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} \mathcal{F}(x_i, W_i)$. Since $\frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} \mathcal{F}(x_i, W_i)$ cannot always be -1, the partial derivative of E with respect to x_l cannot always be 0, which ensures that there will be no gradient vanishing situation.

3.4. Temporal Dependency Modeling and Output

TCN captures temporal features and long-term dependencies in the data through causal convolution, dilated convolution, and residual connections. It enables parallel data processing and allows flexible adjustment of the receptive field size based on the number of layers, kernel size, and dilation factor. Moreover, TCN effectively avoids gradient vanishing and exploding issues compared to traditional RNNs. Since the input of the TCN Block consists of multi-time granularity feature maps with multiple channels, where each channel's data has different features and correlations, the convolution operation can simultaneously process the feature information from multiple channels. Therefore, this paper uses TCN to extract the time features of passenger flow data.

TCN achieves the capture of time series dependencies and avoids the information leakage issue present in traditional convolutional structures through causal convolution. Assuming the output at time t is represented as y_t, it depends only on the input $(x_1, x_2, \ldots, x_{t-1}, x_t)$ before time t and is independent of future inputs $(x_{t+1}, x_{t+2}, \ldots, x_T)$, ensuring the model has strict time constraints.

Although stacking causal convolutional layers can increase the network's receptive field, the longer the length of the sequence that needs to capture the dependencies, the more layers that need to be stacked. This may lead to the issues of gradient vanishing and exploding. TCN addresses these issues by combining causal convolution with dilated convolution, allowing for a significant expansion of the network's receptive field exponentially, without a substantial increase in the number of model parameters [28]. The formula of the dilated convolution operation F on element s of the sequence x is

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{14}$$

where $*$ represents the convolution operation, f represents the filter, d is the dilation factor, k is the filter size, and $s - d \cdot i$ indicates the past direction.

However, even with the use of causal convolution and dilated convolution, the network may still be deep. Therefore, TCN introduces residual blocks to prevent potential information loss and instability issues caused by excessively deep networks.

We assume the shape of the output data of the TCN Block is (b, c, h, w), where b represents the batch size, c represents the number of channels, h is the height of the feature map, and w is the width of the feature map. Since our goal is the single-step prediction of metro passenger flow, we extract the last element from the last dimension of the output data. We reshape the data to (b, h, c) and finally output the predicted results with the shape of $(b, h, 1)$ through a fully connected layer. We can then seek the best model by minimizing the *loss* based on the Mean Square Error (MSE):

$$loss = \text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{15}$$

where N represents the number of samples in the training set, $\hat{y}_i$ is the predicted value of the model, and y_i is the ground truth.

4. Experiment

4.1. Experimental Settings

In this study, all models and data are deployed on the AutoDL cloud server, utilizing the following hardware: RTX 4090 (24 GB) GPU, 12 vCPU Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10 GHz, 90 GB RAM. The software stack includes PyTorch 1.9.0 and CUDA 11.1. We partitioned the training set, testing set, and validation set in a ratio of 6:2:2.

4.2. Parameters

In the Conv Block module, we utilize two (1,1) convolutional kernels, four (1,3) convolutional kernels, and two (1,5) convolutional kernels to extract features. After each convolution, we apply a batch normalization layer and ReLU activation for regularization, resulting in the feature maps with eight channels. In the GCN module, we stack two layers of GCN. During the computation, we maintain the width of the feature maps, performing graph convolution from three perspectives: neighborhood graph, SimRank graph, and cosine similarity graph. This updates the features of each node, which are then concatenated with the original data along the channel dimension. Finally, we use two (1,1) convolutional kernels to expand the channel of the feature maps outputted from the GCN module to eight. The GCN Block employs the Batch Normalization layer (BN) and ReLU activation function for regularization. In the FE Block module, we replace a single-layer convolution with two ordinary residual blocks. The convolutional kernel size of each layer in the residual block is set to (1,3).

The TCN receptive field size is related to the dilation factor b, convolutional kernel size k, and network depth n. Due to the transmission of information between TCN layers using residual blocks, each residual block contains two dilated convolutional layers. Therefore, in each residual block, the receptive field size of TCN can be expressed as $2w - 1$, where w represents the receptive field size of one dilated convolutional layer. Thus, the overall receptive field of TCN can be calculated using the following formula:

$$W = 2 \cdot (1 + \sum_{i=0}^{n-1}(k-1) \cdot b^i) - 1 = 1 + 2 \cdot (k-1) \cdot \frac{b^n - 1}{b - 1} \tag{16}$$

Assuming the length of the historical data is l, the required network depth n to ensure that the receptive field covers all historical data can be calculated using the following formula:

$$n = \left\lceil \log_b\left(\frac{(l-1)(b-1)}{2 \cdot (k-1)} + 1\right) \right\rceil \tag{17}$$

where $\lceil \cdot \rceil$ represents the ceiling function.

The model's additional hyperparameters are fine-tuned using the Optuna framework for optimization, which utilizes the Tree-structured Parzen Estimator (TPE) to iteratively explore and determine the optimal combination of hyperparameters. The model frameworks

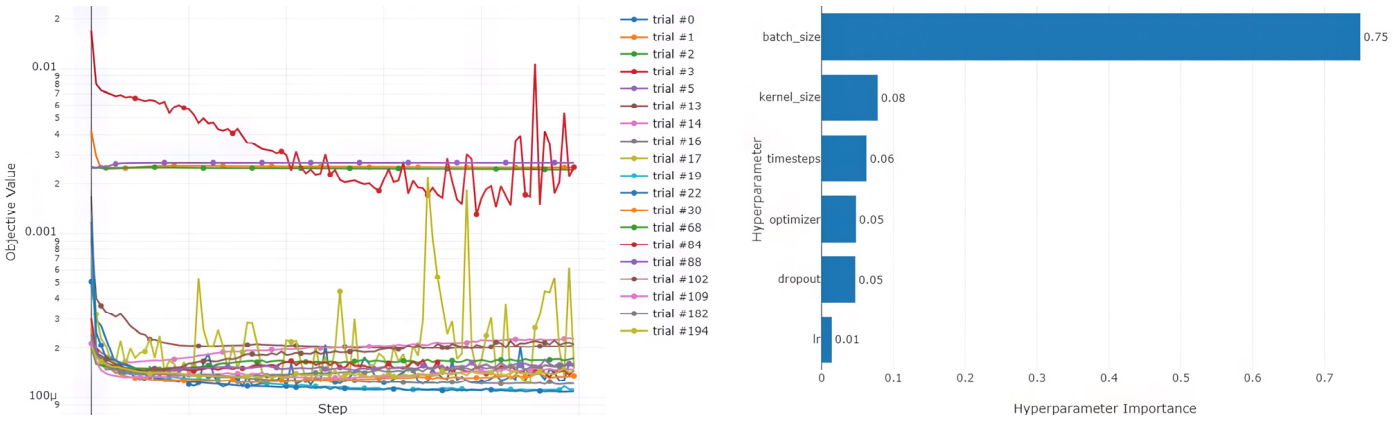
for the 10 min and 15 min target granularities are the same, and the model frameworks for the 30 min and 60 min target granularities can be extended from the model frameworks for these two granularities. Therefore, in this section, the model architectures at the 10 min and 15 min target granularities serve as the foundation for optimizing the hyperparameters of the models. Due to the limitation of the sampling type of hyperparameters in the Optuna framework, the num_channels parameter of the list type in the TCN block cannot be optimized, so in this stage of hyperparameter optimization, we fix the num_channels parameter to (32,24,16,8), the number of model iterations is set to 100. Parameters as well as the range and type of values for each parameter are shown in Table 1. With the objective function of minimizing the model's MSE loss on the validation set, we conducted 200 Trials (we called each iterative training process a Trial) and set up pruning for the hopeless Trial; the best five sets of hyperparameter combinations obtained from the final optimization search are shown in Table 2. The optimization process of each unpruned Trail is shown in Figure 7a, and the importance of each hyperparameter in this optimization process is shown in Figure 7b.

Table 1. Hyperparameter value range.

Parameter Name	Description	Parameter Range	Type
batch_size	the number of data samples captured in one training iteration	(4, 128), step = 4	int
dropout	dropout in the TCN Block	(0.0, 0.99)	float
kernel_size	the filter size of the TCN block	(3, 7), step = 1	int
lr	learning rate	$(1 \times 10^{-5}, 1 \times 10^{-1})$	float
optimizer	optimizer algorithm	[SGD, AdaGrad, RMSprop, Adam]	categorical
timesteps	time step	(8, 32), step = 2	int

Table 2. Hyperparameter optimization results.

No.	Batch_Size	Dropout	Kernel_Size	Lr	Optimizer	Timesteps	Values
1	36	0.3375	3	0.00355	Adam	14	7.08×10^{-5}
2	48	0.1554	4	0.00347	Adagrad	12	7.11×10^{-5}
3	28	0.2917	3	0.00391	Adagrad	8	7.13×10^{-5}
4	16	0.3066	3	0.00397	Adam	14	7.35×10^{-5}
5	12	0.3271	3	0.00334	Adagrad	14	7.78×10^{-5}



(**a**) Optimization Process (**b**) Hyperparameter Importance

Figure 7. Hyperparameter optimization process and importance of hyperparameters. In (**a**), in order to show the differences of the objective values for different Trials, logarithmic values are taken for all the values of the objective value.

Based on the optimal set of hyperparameters in Table 2, we set the batch size of the model to 36, the time step to 14, the learning rate to 0.0036, the optimizer algorithm to Adam, the filter size of the TCN Block to 3, and the Dropout to 0.338. Based on this set of hyperparameters, we optimize the num_channels parameter for the TCN Block. As shown in Figure 8, when the num_channels parameter is set to (32,24,16,8), the model has the smallest MSE on the validation set and is able to completely cover the length of the historical sequences according to the formula of the TCN receptive field. According to the overall architecture of the model, we can set the number_channels parameter of other target granularity models, as shown in Table 3.

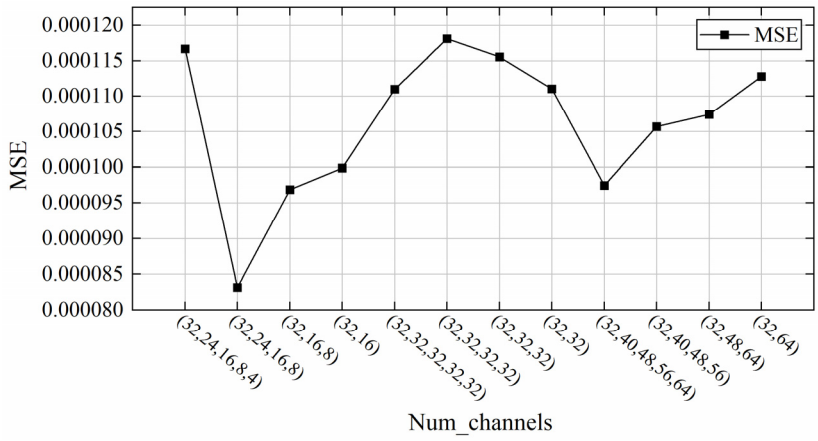


Figure 8. The MSE distribution with different num_channels.

Table 3. The other parameters of the model.

Target Granularity	10 min	15 min	30 min	60 min
Parameter 1	1, 2, 5	1, 3, 5	1, 2, 3, 5, 6, 10, 15	1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30
Parameter 2	20	20	40	60
Parameter 3	32	32	64	96
Hyperparameter 4	(32,24,16,8)	(32,24,16,8)	(64,32,16,8)	(96,48,24,12)

Parameter 1 represents all auxiliary granularities included in the target granularity; Parameter 2 represents the number of convolutional layers in the FE Block; Parameter 3 represents the number of channels of the multi-time granularity feature maps; Hyperparameter 4 represents the number of output channels of each TCN layer.

4.3. Baselines

ARIMA [1]: The ARIMA model is a classic time series analysis and forecasting model. It captures the trend of a sequence using autoregression, differencing, and moving average. It is suitable for stationary time series.

GCN [15]: GCN uses graph convolutional operations to propagate information between nodes, allowing the model to consider relationships between nodes and their neighbors. It is suitable for extracting spatial features from data. However, GCN lacks the ability to model temporal dependencies. Therefore, we combine RNN and GCN to handle metro passenger flow prediction tasks.

GAT [16]: GAT processes graph data using attention mechanisms, introducing attention weights to focus more on important nodes in the graph. It is suitable for handling graph data with different node importance. Similar to GCN, GAT lacks the ability to model temporal dependencies, so we combine RNN and GAT.

TCN [10]: TCN models time series using dilated causal convolutions. Through convolutional operations, it can capture both local and long-term dependencies in a sequence, with a flexible receptive field.

LSTM [8]: LSTM, with its three unique gate structures, captures and utilizes long-term time dependencies effectively. It addresses the issues of vanishing or exploding gradients that RNN faces when dealing with long sequences.

GRU [9]: GRU improves the structure of LSTM by simplifying the gate structures, reducing parameters, and making it easier to train compared to LSTM.

ConvLSTM [29]: The ConvLSTM model combines the characteristics of CNN and LSTM. It has the ability to capture spatial features in data through CNN and model time dependency through LSTM and is suitable for sequence modeling considering both spatial and temporal features.

MTGNN [23]: MTGNN is a powerful graph neural network model for predicting multivariate time series with complex relationships. It integrates graph structure and multi-time scale information, capturing spatial and temporal dependencies through adaptive adjacency matrices, mixed gate mechanisms, graph convolution, and graph attention.

STGCN [22]: STGCN integrates graph convolution and gated temporal convolution through spatio-temporal convolutional blocks, allowing the model to have fewer parameters for faster training, easier convergence, and good flexibility.

We use three widely used evaluation metrics—Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE)—to assess the performance of the model proposed in this paper and compare it with baseline models.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{18}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{19}$$

$$\mathrm{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \times 100\% \tag{20}$$

where N represents the number of predicted values, $\hat{y}_i$ represents the predicted value, and y_i represents the actual value.

4.4. Result and Analyses

4.4.1. Overall Comparison

The performance of the MST-GRT model constructed by us and various baseline models at 10 min, 15 min, 30 min, and 60 min is shown in Table 4.

Table 4. Performance comparison of different methods.

Model	10 min			15 min			30 min			60 min		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
ARIMA	54.18	35.32	24.12%	82.11	50.99	22.97%	161.57	102.27	23.48%	335.46	221.75	25.89%
GCN	48.36	31.88	22.06%	80.82	52.36	24.01%	152.86	96.01	22.04%	337.02	201.76	23.51%
GAT	49.52	29.65	20.31%	81.11	48.94	22.07%	142.60	92.42	21.14%	321.90	197.21	22.98%
TCN	35.08	20.23	13.70%	49.43	30.96	13.77%	92.31	54.14	12.30%	191.74	117.25	13.98%
LSTM	34.17	20.36	13.95%	50.99	29.77	13.43%	96.50	57.26	13.00%	198.01	116.71	13.85%
GRU	34.92	21.07	14.44%	49.40	29.43	13.28%	96.13	56.49	12.97%	217.39	121.36	14.41%
ConvLSTM	33.70	19.85	13.44%	49.97	29.16	13.12%	91.74	53.95	12.02%	208.54	118.14	14.05%
MTGNN	31.33	18.76	12.89%	46.38	27.07	12.18%	81.77	46.78	10.62%	192.26	115.67	13.73%
STGCN	26.92	18.12	12.42%	43.98	25.67	11.58%	82.71	47.28	10.74%	198.46	113.44	13.47%
MST-GRT	24.89	16.34	12.03%	44.10	24.89	11.44%	74.28	45.82	10.23%	171.42	99.62	11.82%

From the results of the comparative experiments, we can draw the following conclusions:

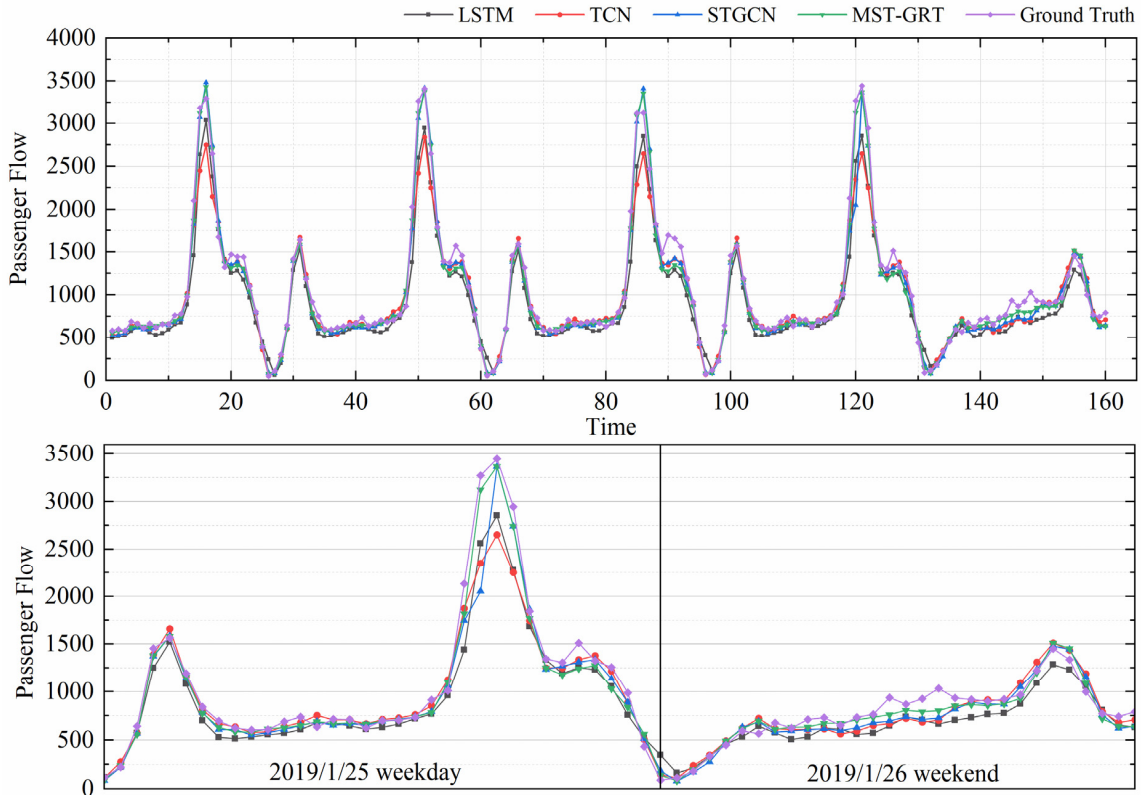
1. The traditional ARIMA model based on statistical theory performs less well than other models in both short-term and long-term aspects. The main reason is its assumption of linear relationships in data, leading to a lack of modeling capability for nonlinear

relationships and an inability to capture long-term time dependencies. Additionally, the ARIMA model fails to consider important topological information in the metro network.

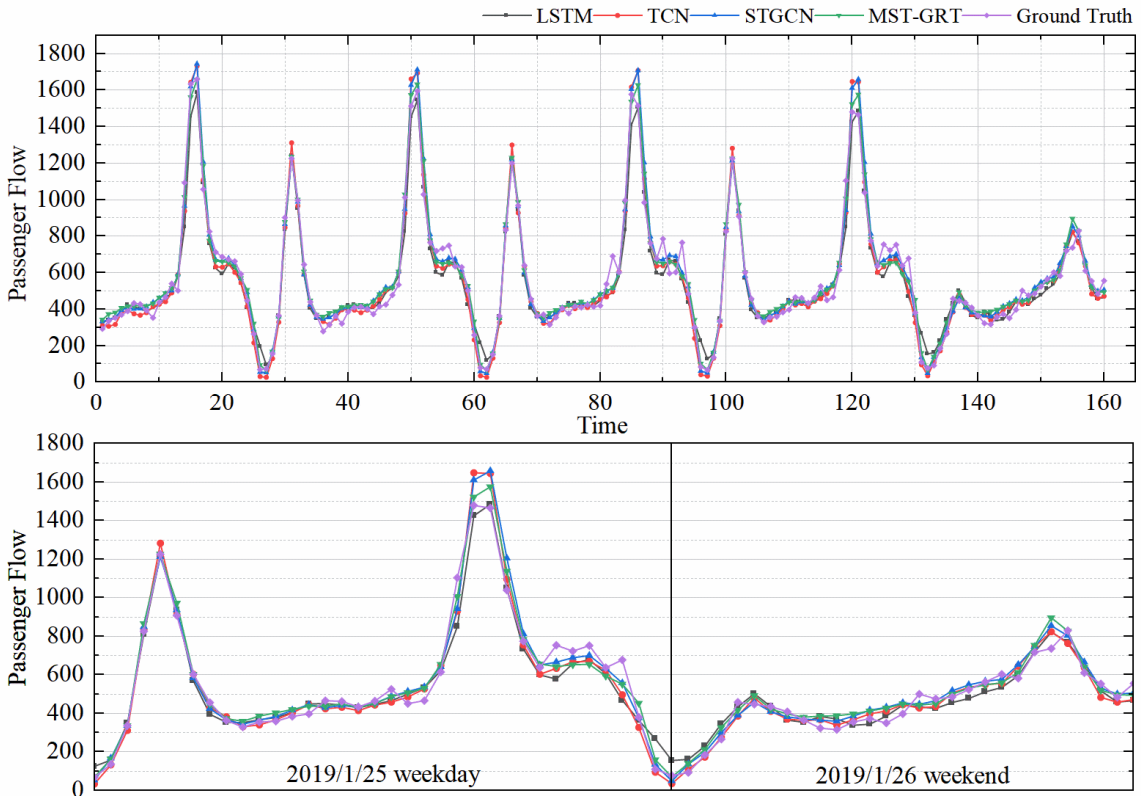
2. GCN and GAT cannot directly capture temporal dependencies in time series data. Therefore, in our experiments, we combined the basic RNN to extract temporal features from the data. From the experimental results, the limitation of RNN in effectively learning long-term dependencies restricts the performance of GCN and GAT models. Even when considering the spatiotemporal features of the data, the models' accuracy is still lower than LSTM and GRU models derived from RNN.
3. TCN and LSTM show similar performance at the 10 min time granularity, with GRU slightly behind. At the 15 min time granularity, all three models show similar performance. The weaker trend and stronger volatility in 10 min time granularity data compared to 15 min time granularity data explain the slightly poorer performance of the GRU model at the 10 min time granularity. At the 30 min time granularity, the TCN model performs the best, possibly due to its fewer parameters compared to LSTM and GRU, making it easier to train and potentially having smoother gradient propagation during training, leading to improved efficiency and performance. At 60 min time granularity, with less data and stronger trends, the larger number of parameters in the LSTM model gives it stronger fitting capabilities, explaining its better performance at larger time granularity.
4. ConvLSTM, MTGNN, and STGCN models can capture both spatial and temporal features in the data. Compared to models that solely capture temporal features like TCN, LSTM, and GRU, their performance is better. However, due to the increased depth of the models, they exhibit overfitting issues at 60 min time granularity, resulting in a significant decrease in prediction accuracy compared to 30 min time granularity. STGCN and MTGNN models both use convolutional operations for extracting temporal features. Compared to RNNs, CNNs have lower dependency on historical data and are more adept at capturing local patterns and features. Therefore, MTGNN and STGCN models outperform the ConvLSTM model at various time granularities.
5. Our MST-GRT model utilizes different graphs to capture information at different scales. It captures local neighbor relationships through the neighborhood graph, structural similarity through the SimRank graph, and passenger flow sequence vector similarity through the cosine similarity graph. This allows the model to better extract spatial correlations between nodes by combining information at different scales. MTGNN relies on an adaptive graph learning module to learn spatial features, while STGCN only considers the neighborhood graph. Therefore, our model's ability to extract spatial features is superior to other models. Additionally, our model uses residual blocks to aggregate data at different time granularities, helping the model better capture changes and trends in time series data. This reduces the model's dependence on a single time granularity, contributing to improved robustness and generalization. Experimental results also demonstrate that our model outperforms other baseline models at various time granularities, showing the best performance at 60 min time granularity.

4.4.2. Results Analysis

Considering the geographical location of each station, we studied the fitting ability of the model proposed in this paper to the passenger flow of the station with more complex functional attributes. We conducted a visualization analysis of the passenger flow prediction results of LSTM, TCN, STGCN, and MST-GRT models at stations 4 and 46, as shown in Figure 9.



(**a**) Station 4



(**b**) Station 46

Figure 9. (**a**,**b**) Represent the predicted results of Station 4 and Station 46, respectively, at the 30 min time granularity.

Station 4 is Jiangling Road Station on Line 1, located in an area with diverse functions, including hospitals, government, businesses, shopping malls, and residential communities. This makes the station an important transportation hub, attracting a large number of passengers. As shown in Figure 9a, all four models can fit the overall trend of passenger flow well, indicating that these models have similar abilities to capture long-term dependencies in the data. However, it can be seen from the details that TCN and LSTM do not fit passenger flow peaks as well as STGCN and the model constructed in this paper. The model constructed in this paper is able to most accurately capture passenger flow peaks and better fit transient changes in passenger flow due to the consideration of passenger flow data at multiple time granularities. However, due to fewer passenger flow data on weekends, the performance of MST-GRT in fitting weekend passenger flow is not outstanding compared to other models, which indicates that the model proposed in this paper requires more training data compared to other models. Station 46 is Qianjiang Road Station, a transfer station for Lines 2 and 4, surrounded by schools, commercial areas, and residential areas. As shown in Figure 9b, the passenger flow at this station changes prominently, with a relatively small difference between the highest and second-highest peaks. It can be seen from the effectiveness of each model in fitting the peak passenger flow that the LSTM performs best, while the TCN performs the worst. Analyzing from the view of model structure, LSTM may be able to better fit this type of data distribution through its gating mechanism, while TCN may lack sufficient model complexity to fit this data distribution. Our model and the STGCN take into account the influence of other stations on the predicted stations, and the spatial and temporal characteristics of the passenger flow at the interchange stations during the peak periods are more complex. The STGCN and MST-GRT may have introduced noise data in the process of graph convolution, which causes the problem of not being able to accurately predict the peak passenger flow; however, since the MST-GRT utilizes a variety of graph structures in graph convolution, it can, to some extent, avoid the interference of noisy data. Thus, our model performs better than the STGCN model. It can also be seen from the heat map of prediction errors at each station (shown in Figure 10) that the model proposed in this paper mostly outperforms the other comparative models in predicting the passenger flow at each station.

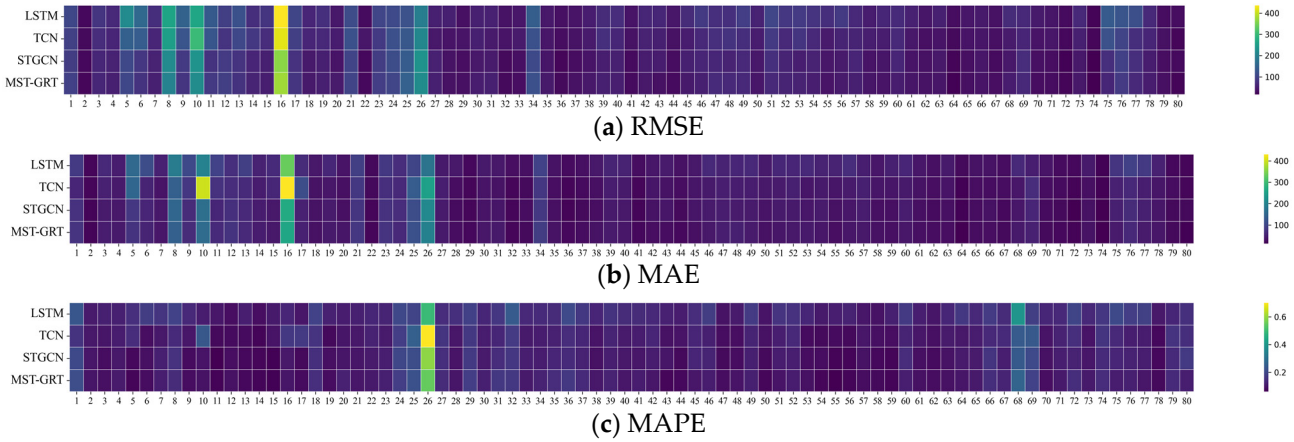


Figure 10. Heat maps of prediction errors for each station.

We explored the influence of passenger flow size on the prediction effect by examining the relationship between the mean passenger flow and each error metric at each station in the test set, as shown in Figure 11. From the figure, we can see that RMSE and MAE are linearly and positively correlated with the mean passenger flow, while MAPE is nonlinearly and negatively correlated with the mean passenger flow. The model proposed in this paper outperforms other models in fitting both smaller and larger passenger flows, especially for stations with smaller average passenger flows, which is attributed to the fact that our model can make full use of the spatial characteristics of metro passenger flow data by considering

the graph structure from multiple perspectives. In addition, because our model is able to supplement the details for larger temporal granularity data by utilizing the data with smaller temporal granularity, it is also able to show better performance when predicting the passenger flow at stations with larger passenger flow. However, as can be seen in Figure 11, the predicted results for the passenger flow of station 26 for each model show an anomaly, which is particularly evident in the MAPE metric.

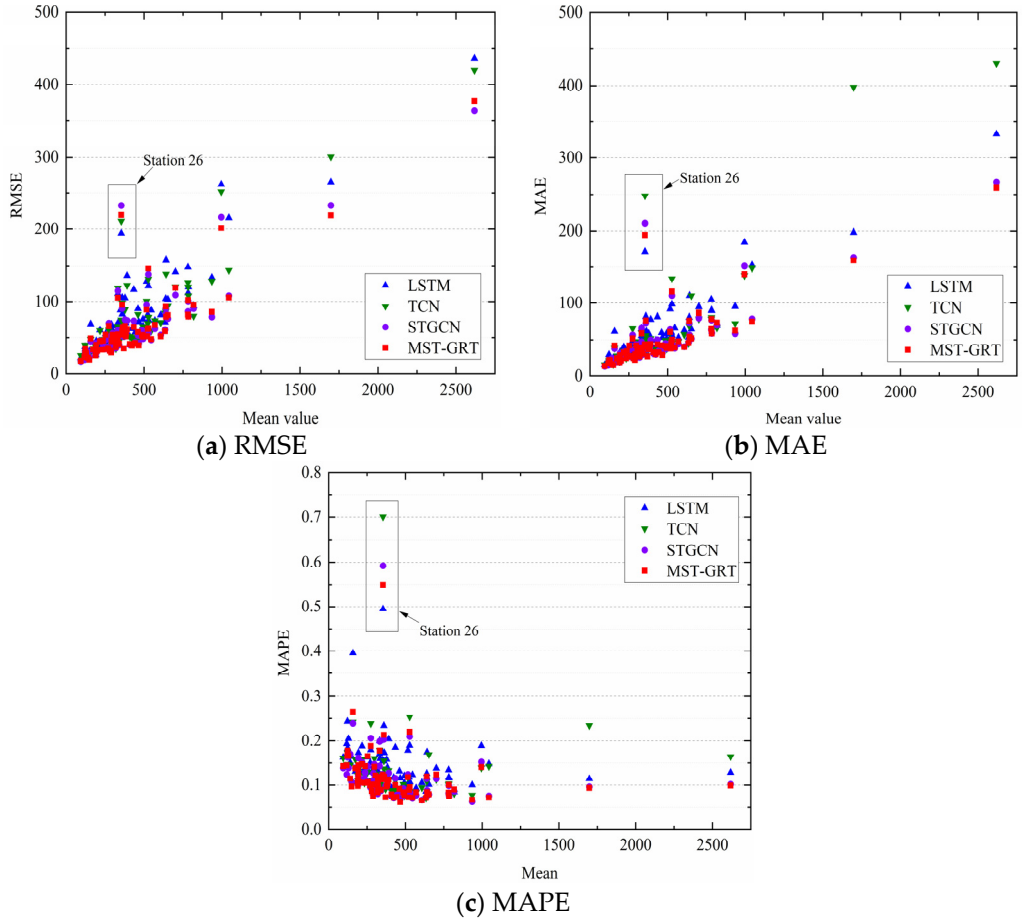


Figure 11. Relationship between station passenger flow and RMSE, MAE, and MAPE.

We selected two stations with the largest MAPE error and two stations with the smallest MAPE error to analyze the reasons for this anomaly. Firstly, we visualized the real values of passenger flow at these four stations and the predicted results of these four stations by the model proposed in this paper, as shown in Figure 12. We used the two-sample Kolmogorov–Smirnov test to analyze whether there is any similarity in the pattern of passenger flow distribution in these four stations, and the results show that the distribution of passenger flow in these four stations is not the same. Then, we used the kernel density estimation (KDE) method to estimate the probability density functions of the passenger flow sequences of the training set and the test set for these four sites, and the results are shown in Figure 13. The figure shows that the distribution pattern of passenger flow at Station 26, which has the largest MAPE error, differs significantly between the training set and the test set. This explains why the predicted results of Station 26 for all the models have a large error. Furthermore, it has been observed that the more similar the distribution of the training data is to that of the test data, the more accurate the model's predicted results. Based on the prediction results of each model for Station 26 in Figure 12, it is evident that the LSTM model outperforms the other models in dealing with anomalies. This reflects the fact that the CNN-based models may not be as effective as the RNN-based models in handling anomalies that differ in distribution patterns between the training data

and the test data. Additionally, the worst performance of the TCN model suggests that utilizing the spatial features of the passenger flow data can slightly reduce the impact of such anomalies on the predicted results.

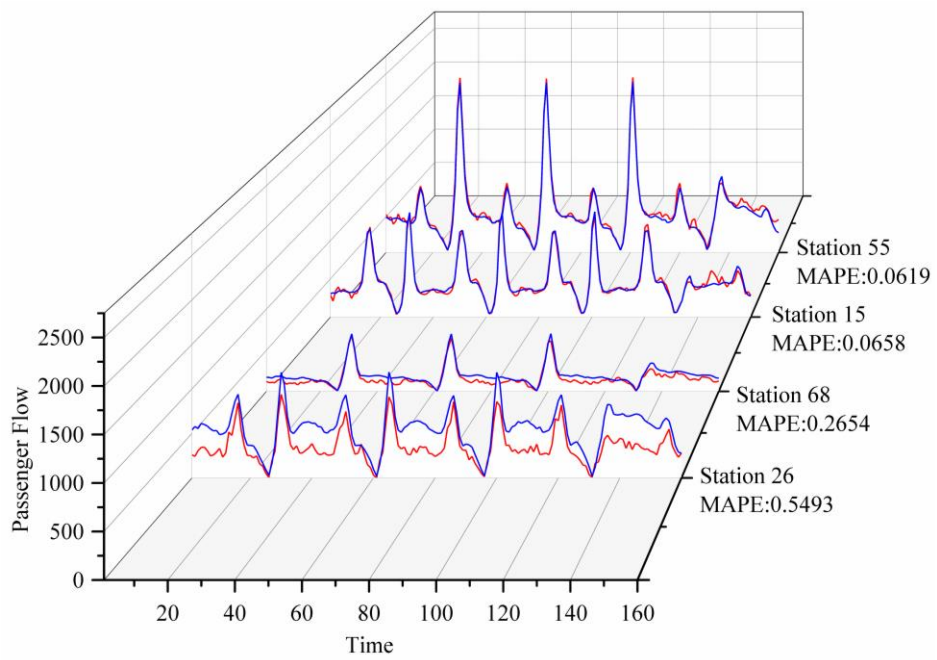
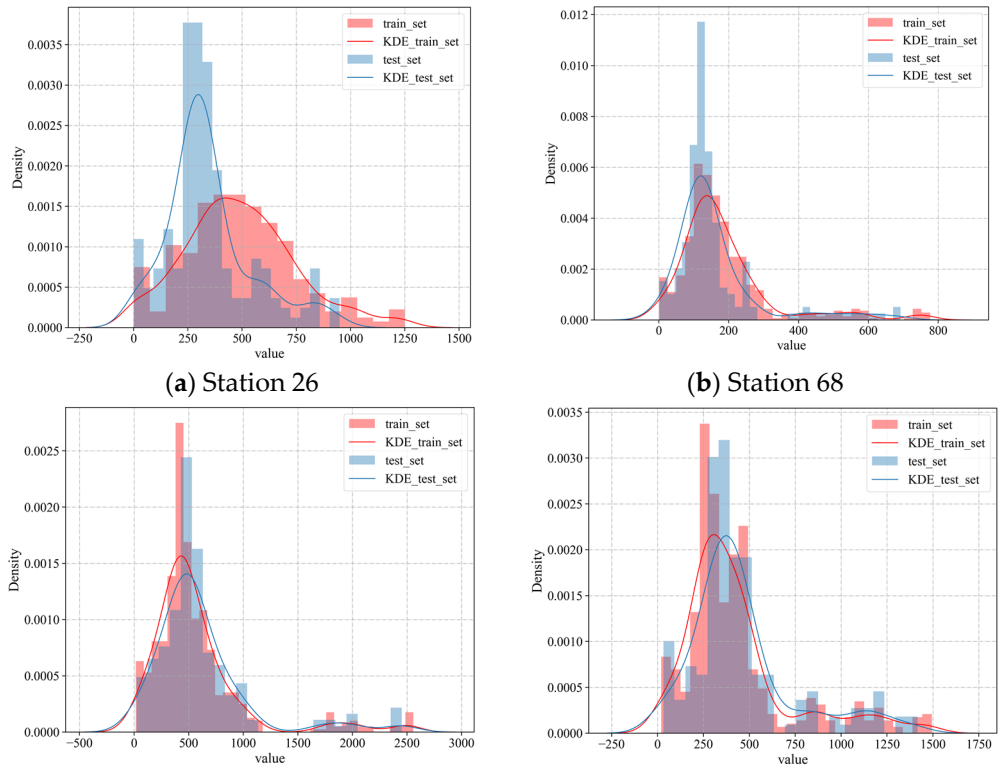


Figure 12. True versus predicted values for the two stations with the largest MAPE errors and the two stations with the smallest MAPE errors, where the blue lines represent the predicted value and the red lines represent the ground truth.



(**a**) Station 26 (**b**) Station 68

(**c**) Station 15 (**d**) Station 55

Figure 13. Kernel density estimation.

Figure 14 illustrates the training process of these four models. After 80 epochs, the training loss curve of our MST-GRT model no longer shows a significant decrease, indicating that the model has converged. This further emphasizes that the MST-GRT model, by considering the spatio-temporal features of the data and aggregating passenger flow data at different time granularities, significantly reduces the loss and achieves faster convergence.

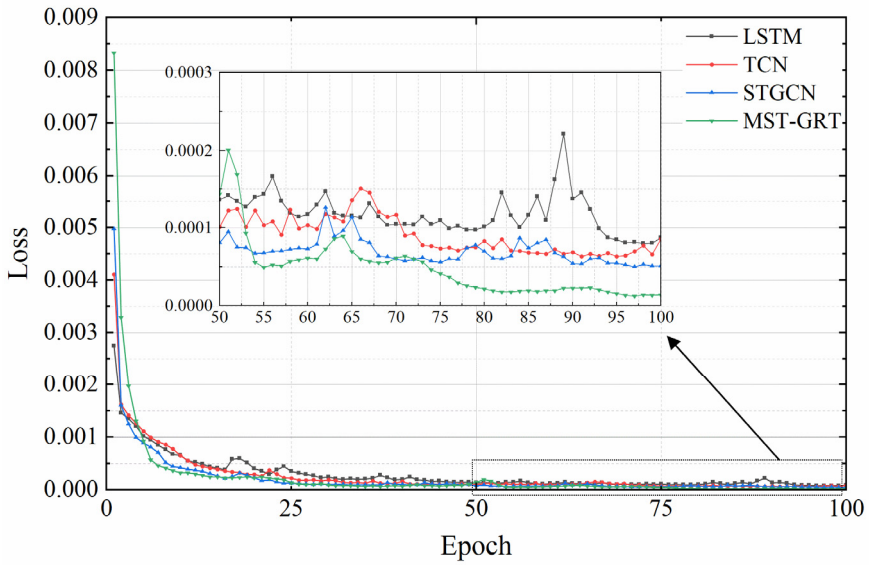


Figure 14. Comparison of training loss.

4.5. Ablation Study

The model we proposed includes the Conv module, multi-graph GCN module, residual blocks, multi-time granularity aggregation module, and 2D-TCN module. In this section, we designed multiple variant models within the same framework and parameters by removing or replacing modules. We explored the performance of different variant models at the 30 min time granularity to validate the effectiveness of each module in our model, and the results are shown in Table 5.

Table 5. Performance comparison of different variant models.

Model	RMSE	MAE	MAPE
V1	96.59	59.03	13.40%
V2	94.65	57.85	13.14%
V3	91.94	54.24	12.11%
V4	99.18	59.69	13.55%
V5	89.32	51.74	11.83%
V6	103.99	61.70	14.17%
V7	81.12	48.23	11.09%
V8	99.60	58.61	13.30%
V9	94.02	55.06	12.50%
V10	87.93	51.68	11.73%
V11	91.37	55.42	12.83%
V12	251.85	176.79	40.16%
V13	198.47	122.74	25.42%
V14	170.57	102.00	23.16%
V15 (Ours)	74.28	45.82	10.23%

V1: This variant model directly uses fully connected layers to replace the residual blocks in the original model for data dimension transformation, aggregating passenger flow data at different time granularities.

V2: This variant model uses a 2D convolutional layer with a filter size of (1,3) to replace the residual blocks in the original model for dimension transformation, aggregating passenger flow data at different time granularities.

V3: This variant model removes the Conv Block for feature extraction of data with the time granularity of less than 5 min.

V4: This variant model removes the multi-graph GCN module for feature extraction of data with the time granularity of greater than or equal to 5 min, while retaining the Conv Block for processing data with the time granularity of less than 5 min.

V5: This variant model only uses neighborhood graphs in graph convolution, while keeping other modules unchanged.

V6: This variant model uses Conv Block for feature extraction for data at all time granularities, while keeping other modules unchanged.

V7: This variant model uses multi-graph GCN for processing data at all time granularities, while keeping other modules unchanged.

V8: This variant model does not use 2D convolution to compress the channel number of the feature maps after aggregating data at each of the two auxiliary granularities, keeping the channel number unchanged after feature extraction through the residual blocks. Other modules remain unchanged.

V9: This variant model, based on V8, doubles the channel number after feature extraction through the residual blocks. Other modules remain unchanged.

V10: This variant model uses 1D convolution to construct TCN while keeping other modules unchanged.

V11: This variant model does not aggregate data at different time granularities, only using data at the target granularity and using 1D convolution to construct TCN. Other modules remain unchanged.

V12: This variant model, based on V11, replaces the TCN constructed by 1D convolution with TCN constructed by 2D convolution.

V13: This variant model uses Attention Feature Fusion (AFF) technology to aggregate data at different time granularities, replacing the concatenation operation in the original model. Other modules remain unchanged.

V14: This variant model replaces the concatenation operation for aggregating data at different time granularities in the original model with a summation operation. Other modules remain unchanged.

V15: The model proposed in this paper.

The time consumption for training each model is illustrated in Figure 15. By analyzing the performance of various variant models, the following conclusions can be drawn: (1) Model V15, which uses residual blocks for dimension transformation, has an accuracy improvement of 24.10% and 22.60% compared to models V1 and V2, which only use fully connected layers and single-layer 2D convolution for dimension transformation. This indicates that the deep network model constructed with residual blocks has stronger expressive power than shallow models. (2) Using multiple filters of different sizes for feature extraction on smaller temporal granularity data allows for capturing detailed features at different scales. This provides a richer complement of details for larger granularity data. This explains why the prediction accuracy of model V3 drops significantly after the removal of the Conv Block. (3) Multi-graph convolution has stronger expressive and generalization capabilities compared to single-graph convolution. Model V5, which uses only the neighborhood graph for graph convolution, has a prediction accuracy decrease of 16.32% compared to model V15. Additionally, model V6, which does not contain a multi-graph GCN module, has a prediction accuracy decrease of 39.33%. This illustrates the importance of node feature aggregation in metro passenger flow prediction tasks. (4) Processing data with multi-graph GCN for all time granularities may result in the loss of original information at small time granularities due to the aggregation of information from related nodes with large feature differences. Therefore, model V7 has a decrease in accuracy of 9.05% compared to model V15. In addition, the training time of model V7

compared to model V15 increased by 28.16%. (5) Feature maps with too many channels may contain redundant features, increasing computational complexity and the risk of overfitting, leading to a decrease in model performance. In feature extraction for various time granularities, expanding and then compressing data channels is necessary to help the model learn richer feature representations. (6) The 2D TCN has advantages over 1D TCN in handling multi-channel passenger flow feature maps. The 2D TCN directly convolves on the feature map, forcing the model to learn the passenger flow features of the entire metro network, while 1D TCN is limited by the format requirements of input data, and the convolution method can only handle passenger flow features of one station at a time. (7) Model V11, considering only target granularity data, has a decrease in prediction accuracy of 26.15% compared to model V15, which integrates data from different time granularities. This indicates that integrating passenger flow data from different time granularities can effectively improve the prediction accuracy of the model. (8) Model V15, which uses concatenation to integrate data from different time granularities, shows significantly improved performance compared to models V13 and V14, which use AFF and summation for integrating. This suggests that concatenation helps the model better understand data features at different time granularities and retains more original information during aggregation, thereby improving model performance.

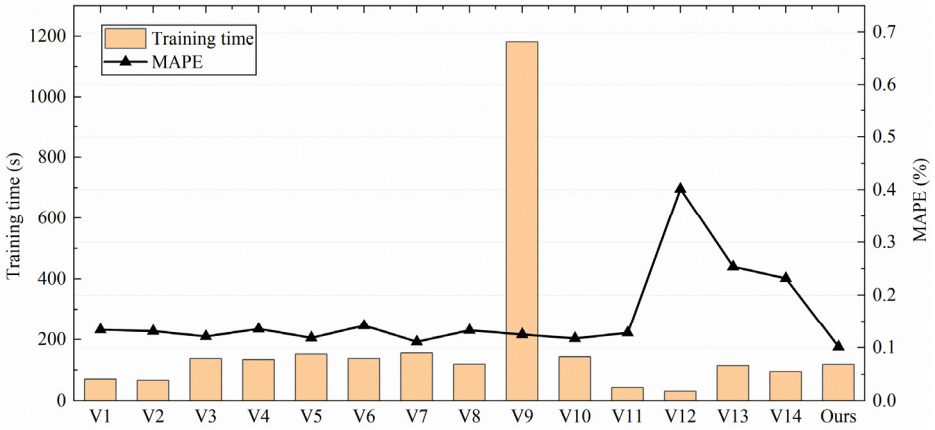


Figure 15. Training time consumption and MAPE error of variant models.

5. Conclusions

The MST-GRT model we proposed aims to fully leverage passenger flow data at various time granularities and the topological structure features of the metro network. It employs multi-graph GCN to deeply explore the spatio-temporal correlations in metro passenger flow data and assists in predicting passenger flow tasks at larger time granularities with data from smaller time granularities. First, we investigated the challenges of merging data at different time granularities and established fusion rules. Next, we studied the correlations of passenger flow sequences at various stations under different time granularities, determined rules for extracting spatio-temporal correlations from different time granularities, and generalized the propagation rules of GCN, which only use neighborhood graphs, to scenarios dealing with multiple graphs. Subsequently, we used residual blocks to aggregate data at different time granularities, constructed multi-channel feature maps, and redesigned TCN with 2D convolution to handle multi-channel feature maps. We then evaluated the performance of the constructed model using the Hangzhou metro smart-card dataset, demonstrating superior performance compared to several baseline models and highlighting the importance of utilizing multi-time granularity passenger flow data and considering various graph structures. Finally, we visualized the predicted results of two metro stations with different functional areas, analyzing the effectiveness and shortcomings of the constructed model. Through ablation experiments, we analyzed the necessity and rationality of each component in the model.

However, the MST-GRT model has limitations: (1) Our model takes longer to train compared to the baseline models. This difference in training time becomes more pronounced as the network depth increases. (2) We did not consider external factors such as weather conditions and metro departure schedules, which may influence metro passenger flow. (3) Our model only selected time granularities that are divisible by the target granularity as auxiliary granularities, without considering other time granularities. In the next steps, we plan to analyze the impact of external factors on metro passenger flow, address the limitations of relying solely on metro passenger flow data and metro topological network structure information by integrating multiple data sources, and develop methods to process unused smaller time granularities data in the model. Optimizing the model with model compression techniques to enhance its generalization and robustness will allow for meeting the standards of transfer learning and applying the model to prediction tasks involving spatio-temporal correlations in data.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L. and C.Z.; software, Y.L.; validation, Y.L. and J.M.; formal analysis, Z.W.; investigation, F.W.; resources, C.Z.; data curation, Y.L. and Y.S.; writing—original draft preparation, Y.L. and S.Z.; writing—review and editing, Y.L. and S.Z.; visualization, S.Z.; supervision, C.Z.; project administration, C.Z.; funding acquisition, C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Jiangsu Transportation Science and Technology Project (2021G09).

Data Availability Statement: The data utilized in this research can be accessed at (https://tianchi.aliyun.com/competition/entrance/231708/introduction, accessed on 1 July 2022).

Acknowledgments: We express our sincere gratitude for the financial support provided by the Jiangsu Transportation Science and Technology Project Fund.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Conejo, A.J.; Plazas, M.A.; Espínola, R.; Molina, A.B. Day-ahead electricity price forecasting using the wavelet transform and ARIMA models. *IEEE Trans. Power Syst.* **2005**, *20*, 1035–1042. [CrossRef]
2. Williams, B.M.; Hoel, L.A. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *J. Transp. Eng.* **2003**, *129*, 664–672. [CrossRef]
3. Chang, H.; Lee, Y.; Yoon, B.; Baek, S. Dynamic near-term traffic flow prediction: System-oriented approach based on past experiences. *Iet Intell. Transp. Syst.* **2012**, *6*, 292–305. [CrossRef]
4. Jeong, Y.S.; Byon, Y.J.; Castro-Neto, M.M.; Easa, S.M. Supervised Weighting-Online Learning Algorithm for Short-Term Traffic Flow Prediction. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 1700–1707. [CrossRef]
5. Wei, Y.; Chen, M.C. Forecasting the short-term metro passenger flow with empirical mode decomposition and neural networks. *Transp. Res. Part C Emerg. Technol.* **2012**, *21*, 148–162. [CrossRef]
6. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef]
7. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training Recurrent Neural Networks. *arXiv* **2012**, arXiv:1211.5063.
8. Kang, D.; Lv, Y.; Chen, Y.y. Short-term traffic flow prediction with LSTM recurrent neural network. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 1–6.
9. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
10. Bai, S.; Zico Kolter, J.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Network s for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271.
11. Dauphin, Y.N.; Fan, A.; Auli, M.; Grangier, D. Language Modeling with Gated Convolutional Networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017.
12. Jiang, W.; Zhang, L. Geospatial data to images: A deep-learning framework for traffic forecasting. *Tsinghua Sci. Technol.* **2019**, *24*, 52–64. [CrossRef]
13. Ma, X.; Dai, Z.; He, Z.; Ma, J.; Wang, Y.; Wang, Y. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. *Sensors* **2017**, *17*, 818. [CrossRef] [PubMed]
14. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 5–10 December 2016.
15. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2017**, arXiv:1609.02907.

16. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *arXiv* **2017**, arXiv:1710.10903.
17. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
18. Wang, S.; Cao, J.; Yu, P.S. Deep Learning for Spatio-Temporal Data Mining: A Survey. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 3681–3700. [CrossRef]
19. Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; Li, H. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 3848–3858. [CrossRef]
20. Li, Y.; Yu, R.; Shahabi, C.; Liu, Y. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. *arXiv* **2018**, arXiv:1707.01926.
21. Li, D.; Lasenby, J. Spatiotemporal Attention-Based Graph Convolution Network for Segment-Level Traffic Prediction. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 8337–8345. [CrossRef]
22. Yu, B.; Yin, H.; Zhu, Z. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Sweden, 13–19 July 2018; pp. 3634–3640.
23. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; Zhang, C.; Assoc Comp, M. Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Virtual, 23–27 August 2020; pp. 753–763.
24. Wu, F.; Zheng, C.; Zhang, C.; Ma, J.; Sun, K. Multi-View Multi-Attention Graph Neural Network for Traffic Flow Forecasting. *Appl. Sci.* **2023**, *13*, 711. [CrossRef]
25. Lu, W.; Zhang, Y.; Li, P.; Wang, T. Mul-DesLSTM: An integrative multi-time granularity deep learning prediction method for urban rail transit short-term passenger flow. *Eng. Appl. Artif. Intell.* **2023**, *125*, 106741. [CrossRef]
26. Jeh, G.; Widom, J. SimRank: A Measure of Structural-Context Similarity. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, AB, Canada, 23–26 July 2002.
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 27–30 June 2016; pp. 770–778.
28. Zhu, J.; Su, L.; Li, Y. Wind power forecasting based on new hybrid model with TCN residual modification. *Energy AI* **2022**, *10*, 100199. [CrossRef]
29. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-K.; Woo, W.-C. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *arXiv* **2015**, arXiv:1506.04214.