Multi-Spatial Scale Spatio-Temporal Transformer: A Refined Traffic Data Forecasting Method

China University of Mining and Technology

Lei Zhang (zhanglei@cumt.edu.cn)
China University of Mining and Technology

Bailong Liu
China University of Mining and Technology

Zhizhen Liang
China University of Mining and Technology

Xuefei Zhang

Research Article

Keywords: Traffic flow prediction, spatial-temporal dependency, dynamic graph neural networks, Transformer

Posted Date: July 26th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3190420/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

Additional Declarations: No competing interests reported.
Multi-Spatial Scale Spatio-Temporal Transformer: A Refined Traffic Data Forecasting Method

Yue Zhang\textsuperscript{1,2}, Lei Zhang\textsuperscript{1,2*}, Bailong Liu\textsuperscript{1,2}, Zhizhen Liang\textsuperscript{1,2}, Xuefei Zhang\textsuperscript{3}

\textsuperscript{1}Engineering Research Center of Mine Digitalization (Ministry of Education), China University of Mining and Technology, Xuzhou, 221116, Jiangsu, China.
\textsuperscript{2}School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, 221116, Jiangsu, China.
\textsuperscript{3}Jiangsu Hengwang Digital Technology Co., Suzhou, 215000, Jiangsu, China.

*Corresponding author(s). E-mail(s): zhanglei@cumt.edu.cn;
Contributing authors: zhang_yue@cumt.edu.cn; liubailong@cumt.edu.cn; liang@cumt.edu.cn; 125619789@qq.com

Abstract

Accurate traffic data forecasting is essential to improve the efficiency of intelligent transportation systems. Existing traffic prediction models only model spatial dependency based on the connectivity of roads, which overlooks the characteristic information of hidden spatial dependency and leads to a loss of prediction accuracy. In addition, there exists a strict relative positional relationship in the temporal dependency between traffic data, which is often overlooked by existing models, making it difficult to accurately model the temporal dependency. To solve these problems, this paper proposes a traffic data prediction method (MSS-STT) based on Multi-Spatial Scale Spatio-Temporal Transformer. MSS-STT first employs multiple specialized spatial Transformer networks to model different spatial scales in order to capture spatial dependencies and patterns at various levels. It also utilizes graph convolutional neural networks to extract static spatial structural features. Then, a gating mechanism is used to fuse the spatial dependencies from different spatial scales and the static spatial features. Finally, MSS-STT extracts different temporal dependencies by considering the order of time points and the varying contributions to the prediction from different relative positions between time points in historical traffic data. Experiments on three real-world
datasets from the Caltrans Performance Measurement System (PeMS) show that the proposed MSS-STT model outperforms the state-of-the-art methods.

**Keywords:** Traffic flow prediction, spatial-temporal dependency, dynamic graph neural networks, Transformer

1 **Introduction**

Traffic data prediction is a fundamental task in the field of spatio-temporal data mining, where the goal is to utilize historical traffic data from the current node and its neighboring nodes to forecast the future traffic conditions at that node (e.g., speed, flow, and density). Accurate traffic data prediction holds significant value and importance in numerous applications within Intelligent Transportation Systems (ITS) [1]. For example, it enables proactive intervention and active control of traffic based on data forecasts, thereby mitigating traffic congestion issues. Traffic data are usually presented in the form of spatio-temporal graphs, and the spatial dependencies at different time points in such spatio-temporal diagrams are dynamically changing. Modeling the multidimensional and complex dependencies in traffic data is the key to solving the traffic data prediction problem. In the early days, to model complex, highly dynamic, and nonlinear spatio-temporal data, researchers applied traditional time series models to the problem. Then, with the development of artificial intelligence, some classical algorithms of machine learning and deep learning were gradually used to predict traffic data and achieved good results. Nowadays, with the development of graph neural networks (GNNs) [2], more and more traffic prediction models are devoted to modeling the spatial dependence of traffic data using graphs, and some models based on GNNs and their variants have shown better performance than previous methods in various tasks of traffic data prediction. However, the spatial dependence of the traffic system is more complex, and the temporal dependence between traffic data is largely influenced by the relative location relationship between time points, so the traffic prediction problem is still limited and challenged by the following aspects.

(1) **Comprehensive modeling of spatial dependencies:** The spatial dependency of urban transportation networks is more complex, including static spatial structure caused by road connections and hidden spatial dependency caused by road attributes and regional functions. These factors together influence the structure and behavior of the traffic network and have an important impact on traffic characteristics such as traffic flow, congestion, and traffic speed. Therefore, an accurate understanding and modeling of the spatial dependencies of urban transportation networks requires a comprehensive consideration of multiple scales to reveal the complex spatial dependency patterns in transportation networks. Previous traffic prediction models would define logical connectivity between sensor nodes and then extract spatial dependencies through this connectivity. For example, DCRNN [3] and ASTGCN [4] define logical connectivity relationships between sensors in terms of distance thresholds. DCRNN [3] uses a diffusion process to perform graph convolution operations, and ASTGCN

2
uses graph convolution and attention mechanisms to obtain global spatial dependencies. Although these methods achieve good results, there is a certain subjective factor in the selection of the threshold, which can prevent the model from extracting spatial dependencies accurately and purposefully. Instead of predefining the connectivity of sensor nodes, Graph WaveNet uses an adaptive graph learning algorithm to extract the hidden connectivity between sensor nodes, but this adaptive matrix is fixed after training and cannot be dynamically adjusted with the data characteristics. To address this issue, some models such as AdapGL, TVDBN proposed the concept of dynamic graphs, i.e., constantly changing the graph structure to represent highly dynamic spatial dependencies at different times. But this frequent construction of new traffic graphs may increase the computational and storage burden. Models such as STTN and S2TAT extract spatial dependencies based on a Transformer network by considering sensors as all connected and constructing traffic graphs from them. Although this approach can fully extract spatial dependencies, it causes the problem of over-extraction, which leads to the introduction of more noise and a large amount of useless information. All of the above approaches focus only on the connected relationships between sensor nodes and ignore the influence of other scales (e.g., road attributes where sensors are located, area function, etc.) on traffic data. Road attributes (e.g., speed limit information, number of lanes, road type, etc.) have an important impact on traffic data, and sensor nodes in the same road attribute will have similar features even if they are physically distant. Failure to consider feature information at the road attribute level can result in incomplete spatial dependence modeling and thus loss of prediction accuracy. In addition, area features have an equally important impact on traffic data. As shown in Fig. 1 and 2, the yellow line indicates the speed data returned by a sensor on a day, and the red line is the speed data returned by a sensor closer to it on the same day. These two lines correspond to the sensors that belong to the same A area. The green line corresponds to the speed data returned by the sensor farther away from the yellow line on the same day, which belongs to the B area. It can be seen that there is a clear heterogeneity between different regions and that traffic characteristics can vary widely in different areas during the same time. If the feature information at the level of regional function is not considered, it will lead to the inability to extract accurate and comprehensive spatial dependencies.
Fig. 1 The three lines indicate the speed profiles of different sensors on the same day. The red and yellow dashed lines indicate the data coming back from the sensor located in area $A$, and the green line indicates the data coming back from the sensor located in area $B$.

Fig. 2 The specific location of the three sensors, red, yellow, and green dots correspond to the red, yellow, and green lines in Fig.1 respectively

(2) Optimal modeling of temporal dependence: Temporal dependence, spatial dependence, and spatio-temporal dependence need to be considered together in traffic data prediction. Take the spatio-temporal graph shown in the Fig.3 as an example, spatial dependency means that different sensors will influence each other in the
same time. Temporal dependence means that for the same sensor, the eigenvalues of a certain time step are closely related to the previous time steps. For example, $A''$ in the Fig.3 will be influenced by point $A$ and $A'$ of the previous moment. The spatio-temporal dependence refers to the fact that the traffic characteristics of a sensor at a certain time step are influenced not only by the previous time step characteristics of that sensor but also by the previous time step characteristics of its neighboring sensors. As shown in the Fig.3, $A''$ will be influenced by points $B$, $C$, $B'$ and $C'$.

![Fig. 3](image)

**Fig. 3** Complex dependencies contained in traffic data, including temporal dependencies, spatial dependencies, and spatio-temporal dependencies

In the early days, most models considered only spatial and temporal relations and used asynchronous modeling to describe these two relations. Examples include DCRNN [3], ASTGCN [4] and STGCN [11]. These models use two separate components to capture spatial dependence and temporal dependence in turn. Then the post-extracted temporal correlation uses the data after capturing the spatial dependence, not the original data, which can lead to inaccurate extraction. Subsequently, STSGCN [12] and STFGNN [13] proposed local spatio-temporal graphs to implement simultaneous spatio-temporal modeling. This model only considers local spatio-temporal relationships and achieves good results in short-term prediction, but it still has major problems in long-term modeling. In traffic data prediction, historical traffic data usually consists of data from a series of time points, such as traffic flow, speed, or congestion in the past period. The order of these time points and their different relative positions to each other lead to differences in the degree of time dependence. For example, for a point in time in traffic data, the data from its previous point in time will have a greater impact on it than the data from the previous two points in time. In addition, the data at the current point in time is not affected by the data at the later point in time. However, existing models do not consider this aspect, so how to optimize the modeling temporal dependence is also a major problem to be solved.

To deal with the limitations of existing models, we introduce a new method, dubbed the Multi-Spatial Scale Spatio-Temporal Transformer (MSS-STT), to efficiently solve the traffic data prediction problem. In MSS-STT, the Transformer-based Multi-scale Spatial dependency aggregation (TMS) module is proposed. This module uses multiple specific spatial Transformer networks to extract dynamic spatial dependent features
at different scales. It uses graph convolution networks to learn static spatial structure features and a gating mechanism to fuse each spatial structure feature according to its importance to the prediction. Different from the previous spatial feature extraction module, TMS uses a multi-scale feature extraction approach to target different levels of influencing factors. Thus, a more accurate and comprehensive spatial dependence representation is obtained, and the purpose of refined modeling of spatial dependence is achieved. Moreover, instead of extracting spatial-temporal dependencies separately, we embed the data after extracting spatial dependencies into the corresponding time points. Subsequently, we perform temporal dependencies feature extraction on the time series consisting of several time points. The temporal dependencies extracted in this way are considered to contain non-local spatio-temporal dependencies. In order to solve the problem of losing relative position information when extracting temporal dependencies, the Traffic TimeWise feature extraction (TTW) module is designed in MSS-STT. TTW achieves selective extraction of temporal dependent features based on the traffic characteristic that current traffic data are not influenced by future traffic data and the factor that different relative location information of the time series composed of traffic data have different contributions to the prediction. Overall, our contributions are summarized as follows:

(1) The TMS module is proposed to extract spatial structure features from three scales: sensor node level, road level, and area level, respectively. Among them, for the road level and area level extraction, corresponding filters are designed to model spatial dependencies in a targeted manner. In addition, Graph Convolutional Network (GCN) is used in this module to extract static spatial structure features. Finally, Gated Selection Aggregating (GSA) is designed to aggregate individual spatially dependent features in TMS using a gating mechanism. Compared with existing methods, this module can extract spatial features at each different scale in a targeted manner. A comprehensive and refined extraction of spatial structures is achieved.

(2) The TTW module is proposed to capture the temporal dependence of traffic data with greater accuracy. Instead of extracting spatio-temporal dependencies separately, this paper uses the TMS module to first extract dynamic spatial dependency features and subsequently embed these spatial features into the corresponding time points, so that each time point in the traffic data has different spatial dependencies. After that, temporal-dependent features are extracted for a time series consisting of several time points. We argue that the temporal-dependent features extracted in this way are those that contain non-local spatio-temporal dependencies. More importantly, the module takes into account the relative weight information from the relative positions between time points in the time series. This allows for the selection of more valuable historical data to be utilized in the extraction of time dependence. The module can better reflect the differences in the degree of temporal dependence between different relative positions, allowing the model to better adapt to the actual traffic data characteristics.

(3) Extensive experimental results on three datasets show that the proposed model MSS-STT outperforms the baseline, proving that our model possesses better traffic data prediction capability.
2 Related Works

2.1 Prediction Based on Traditional Models

Traffic data prediction [14] is a typical spatio-temporal data prediction problem, which uses the historical traffic data of a node itself and its neighboring nodes to predict the future characteristic data of that node.

Many researchers have made great efforts to achieve more accurate and efficient traffic flow forecasting. In the early days, the traffic forecasting problem was commonly analyzed by traditional time series models. For example, the autoregressive integrated moving average model (ARIMA) [15] and the Kalman filter model [16], have achieved good results in short-term forecasting problems. However, since the traditional time series model can only capture linear relationships and not nonlinear relationships, it is only applicable to stable traffic flows. It is no longer applicable if the traffic conditions change drastically. Later, traditional machine learning methods were applied to this field to model more complex data. These methods are characterized by high data processing power, flexible implementation, and high generalization ability, making the predictable range a bit wider. However, traditional machine learning methods have a limited potential learning space, limiting their ability to extract features from big data. Therefore it is difficult to take into account the spatio-temporal correlation of high-dimensional traffic data.

2.2 Prediction Based on Convolutional Neural Networks

With the widespread use of Convolutional Neural Networks (CNNs) [17, 18], they have shown powerful performance in the field of traffic prediction. The reason is that CNNs have excellent feature extraction capability. To capture spatio-temporal correlation, many deep learning-based traffic prediction methods have emerged. Traffic prediction has been further advanced by modeling the entire city as a grid and using CNN for feature extraction. Zheng et al. proposed ST-ResNet [19]. ST-ResNet employs a Convolutional Neural Network to model the correlation between adjacent regions and uses residual connectivity to alleviate the overfitting problem in spatio-temporal prediction. Yao et al. proposed DMVST-Net [20], which uses both CNN and long short-term memory networks (LSTM) to capture complex spatio-temporal relationships. Wang [21] et al. proposed MAGCN, MAGCN to divide the city into a grid of unequal size based on the attributes of the region. The regional traffic is then predicted using a matrix constructed using the traffic based on the Origin-Distribution of functional areas. These models improve the performance of prediction to some extent, but all of them are applied to Euclidean data (i.e., images, text, and video). In real-life scenarios, traffic flow data is non-Euclidean data and often has complex dependencies. Therefore, dividing cities into regular grids and using Convolutional Neural Networks to process spatio-temporal data is not the optimal traffic prediction method.
2.3 Prediction Based on Graph Neural Network

In recent years, Graph Neural Networks (GNNs) [22, 23] have become the frontier of deep learning research and have shown state-of-the-art performance in various applications. However, the early processing of graph data involved representing the graph structure data as a set of vectors in the preprocessing stage of the data. This method has obvious drawbacks for some graph-rich data. A lot of structural information will be lost and the prediction accuracy will heavily depend on the pre-processing of the graph data. After that, Marco Gori [24], Franco Scarselli [25, 26], and others proposed the concept of a Graph Neural Network to model graph data. GNN is proposed to architect the data processing process directly on top of the graphical data. This not only extends the existing neural network models but also greatly improves the accuracy of processing graph data. GNN initially focuses on RNN as the main framework to generate vector representation for each node by simple feature mapping and node aggregation. However, this cannot cope with the complex and variable graph data in reality. To address this situation, Bruna et al [27] proposed to apply CNNs to Graph Neural Networks. By a clever transformation of the convolution operator, they proposed the frequency domain-based and space domain-based GCNs and derived many variants. This reduces the complexity of the computation of Graph Neural Network models and makes the computation of Laplace matrices in the computation of graph networks a thing of the past. GraphSAGE [28] was proposed, and GraphSAGE used the sampling mechanism to handle larger graphs and to obtain representations of new nodes for unknown ones. Graph Attention Network (GAT) [29] was born to specifically address the problem in which different neighbor nodes have different weights during GNN aggregation of neighbor nodes. The Graph Attention Network borrows the attention mechanism from the Transformer model. In the analysis of graph data, different weights can be assigned to neighboring nodes according to their characteristics. GAT does not need to know the entire graph structure to train GCN. But it explores the neighboring nodes of each node and the computation is fast enough. Parallel computations can be performed on different nodes and unsighted graph structures can be processed in both supervised and unsupervised ways.

For traffic prediction, the most important thing is the processing of traffic graph data [30]. In order to model spatial dependencies more effectively, more and more models are applying graph neural networks to the field of traffic prediction. Li et al. proposed the diffusion convolutional recurrent neural network (DCRNN) [3], which uses diffusion graph convolutional network and GRU to adapt the direction of traffic flow and learn the representation of spatial dependencies and temporal relationships. However, this model extracts only static spatial structures in extracting spatial dependencies, which is not reasonable for traffic data prediction. Subsequently, Yu et al. proposed STGCN [11] and Wu et al. proposed Graph WaveNet [5]. Both of them used graph convolution on the spatial domain to model the traffic graph, using temporal convolution layers instead of RNN for temporal modeling thus speeding up the training phase of the model. Where STGCN uses spectral convolution defined on undirected graphs to model spatial correlations. Graph WaveNet constructs an adaptive matrix to account for variations in the influence between nodes and their neighbors, automatically discovering invisible graph structures from the data, but this adaptive
matrix is fixed after training and cannot be dynamically adjusted with data characteristics. The use of static spatial structure extraction is not sufficient for modeling highly dynamic traffic data. So using GCN alone is not sufficient to accurately model dynamic spatial dependencies.

### 2.4 Prediction Based on Attention and Transformer

Most previous models use recurrent neural networks (RNNs) and their variants to extract temporal dependencies. For example, Hu et al. [31] proposed a variant long short-term memory (LSTM) based time series prediction method. Although this method has a good performance in short-term traffic data prediction, the method using RNN and its variants will be limited in extracting long-term time dependence.

Attention models (AMs) [32, 33] have the advantages of low complexity and fewer parameters compared with CNNs and RNNs. At the same time, AMs solve the problem that RNNs cannot be computed in parallel. AMs do not depend on the computation result of the previous step at each step, so they can be processed in parallel as CNNs. In addition, AMs can catch the focus from long texts without losing important information. Solving the problem that long-distance information is weakened. In order to accurately model the dynamic spatial dependence of traffic data, the attention-based spatio-temporal Graph Convolutional Network ASTGCN [4] is proposed. The model introduces two attention layers in STGCN [11] to capture dynamic dependencies in spatial and temporal dimensions, respectively, and ASTGCN [4] achieves good results. However, ASTGCN [4] and STGCN [11] ignore the synchronization of spatio-temporal data. They both use two different components to capture spatial dependence and temporal dependence, respectively. This separate extraction method fragments the spatio-temporal relationship to a certain extent. Unlike them, Bai et al. designed STG2Seq [34], which attempts to model spatio-temporal correlations simultaneously by using a gated residual GCN module with two attention mechanisms. However, the connectivity characteristics of the nodes at different time steps mask the spatio-temporal correlations to the extent that the heterogeneity of spatio-temporal data cannot be captured. Song et al. proposed STSGCN [12], a model that can effectively capture complex local spatio-temporal correlations through a well-designed spatio-temporal synchronization modeling mechanism. Meanwhile, multiple modules with different time periods are designed in the model to effectively capture the heterogeneity of the local spatio-temporal graph. The local spatio-temporal graph has some limitations and is not effective in long-term forecasting tasks. Later, Xu et al. proposed a new variant of GNNs based on Transformer [8], namely the spatio-temporal Transformer[10]. It jointly utilizes dynamic directional spatial dependence and long-term temporal dependence to improve the accuracy of long-term traffic flow prediction. The combination of spatio-temporal Transformer and traffic prediction makes the prediction accuracy of this problem. The combination of spatio-temporal Transformer and traffic prediction leads to a certain improvement in the prediction accuracy of this problem and a more rational approach.
### 3 Proposed Solution

#### 3.1 Problem Definition

In this section, we define some of the symbols used in this paper and formally define the setup of our problem. The list of symbols is shown in Table 1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G)</td>
<td>Spatial graph</td>
</tr>
<tr>
<td>(V)</td>
<td>Set of nodes in graph</td>
</tr>
<tr>
<td>(E)</td>
<td>Set of edges in graph</td>
</tr>
<tr>
<td>(N)</td>
<td>Number of nodes, (N =</td>
</tr>
<tr>
<td>(W)</td>
<td>Graph adjacency matrix</td>
</tr>
<tr>
<td>(x_i)</td>
<td>The velocity value of node (i) at times (t)</td>
</tr>
<tr>
<td>(X_{in})</td>
<td>The velocity value of all nodes at times (t)</td>
</tr>
<tr>
<td>(T)</td>
<td>Length of historical time series</td>
</tr>
<tr>
<td>(T_e)</td>
<td>Length of prediction time series</td>
</tr>
<tr>
<td>(\hat{Y})</td>
<td>Prediction</td>
</tr>
<tr>
<td>(W)</td>
<td>Learnable model parameters</td>
</tr>
</tbody>
</table>

**Definition 1:** In order to better monitor the traffic conditions in a city, roads are usually equipped with sensors to collect characteristics such as vehicle flow and speed on that road. We define the traffic graph with sensors as nodes, denoted as \(G = (V, E, W)\), where \(V\) denotes the set of sensor nodes, \(N = |V|\) denotes the number of sensor nodes, \(E\) is the set of edges reflecting the logical connectivity between sensor nodes, and \(W \in \mathbb{R}^{N \times N}\) is the adjacency matrix constructed with the Euclidean distance between sensors via a Gaussian kernel.

**Definition 2:** \(x_i \in \mathbb{R}\) denotes the velocity characteristics of the sensor node \(i\) at the time step \(t\), \(x_t = (x_1^t, \ldots, x_i^t, \ldots, x_N^t) \in \mathbb{R}^{N}\) denotes the velocity characteristics of \(N\) sensor nodes at the time step \(t\), and \(X_{in} = (x_{t-T+1}, \ldots, x_t) \in \mathbb{R}^{N \times T}\) denotes the traffic characteristics of \(N\) sensor nodes at the \(T\) time step.

**Definition 3:** The traffic data prediction task is defined as inputting the traffic features \(X_{in}\) of \(N\) sensor nodes in the past \(T\) time steps, training a prediction function \(f\), and effectively infer the traffic features \(\hat{Y} \in \mathbb{R}^{N \times T_e}\) in the future \(T_e\) time steps, denoted as: \(X_{in} \xrightarrow{f} \hat{Y}\). In this paper, we focus on the prediction of velocity characteristics, and the calculation of flow and occupancy is similar to that of velocity characteristics.

#### 3.2 Overall Architecture

As depicted in Fig. 4, the Multi-Scale Spatial-Temporal Transformer network consists of a Transformer-based Multi-scale Spatial dependency aggregation (TMS)
module, a Traffic TimeWise feature extraction (TTW) module, and a prediction module. The TMS module includes an area spatial dependency extraction layer, a road spatial dependency extraction layer, a static spatial structure feature extraction layer, and gated selection aggregating layer. The TMS module uses multiple specific Transformer networks to model spatial dependencies at different levels in a targeted manner through multi-scale feature extraction, achieving the purpose of refined modeling of spatial dependencies. The TTW module can selectively extract temporal-dependent features based on the different contributions of traffic characteristics and relative location information in the time series to the forecast. Finally, the prediction module uses a convolution operation to predict future traffic data based on the obtained spatio-temporal features. In order to learn the deeper features, the model forms a block with TMS and TTW, cycling $N_{\text{blocks}}$ times. The computational procedure of MSS-STT is shown in Algorithm 1.

![Algorithm 1](algorithm.png)

**Fig. 4** The MSS-STT framework diagram; in the framework diagram, the red circle indicates sensor $A$. Taking sensor $A$ as an example, the input is the historical data of $T$ time steps, which first passes through a convolutional layer to expand the feature dimension, and then passes through the spatially dependent feature extraction layers at the node, area, and road levels and the static spatial structure feature extraction layer, respectively, and then passes through the GSA for aggregation. TMS and TTW form a block that loops $N_{\text{blocks}}$ times. And finally the data obtained by looping $N_{\text{blocks}}$ times is fed into the prediction module to predict the velocity data after the $T_*$ time step.
Algorithm 1 Algorithm in MSS-STT

Input: $X_{in} \in \mathbb{R}^{N \times T}$, $W \in \mathbb{R}^{N \times N}$
Output: $\hat{Y} \in \mathbb{R}^{N \times T}$

1: $X \in \mathbb{R}^{C \times N \times T} \leftarrow \text{Conv}(X_{in})$  \hspace{1cm} $\triangleright$ Expanded Feature Channel
2: $X' \in \mathbb{R}^{C \times N \times d_G \times T} \leftarrow (X, R_{area} \in \mathbb{R}^{N \times N})$  \hspace{1cm} $\triangleright$ Position embedding
3: for $n_{block} = 1$ to $N_{blocks}$ do
4: \hspace{1cm} $S_{node} \leftarrow X$  \hspace{1cm} $\triangleright$ Feature extraction at the node level
5: \hspace{1cm} $S_{area} \leftarrow \text{Transformer-Area}(X)$  \hspace{1cm} $\triangleright$ Feature extraction at the area level
6: \hspace{1cm} $S_{road} \leftarrow \text{Transformer-Road}(X)$  \hspace{1cm} $\triangleright$ Feature extraction at the road level
7: \hspace{1cm} $S_{static} \leftarrow \text{GCN} (X, W \in \mathbb{R}^{N \times N})$  \hspace{1cm} $\triangleright$ Feature extraction for static spatial structures
8: \hspace{1cm} $X_S \leftarrow \text{GSA} (S_{node}, S_{area}, S_{road}, S_{static})$  \hspace{1cm} $\triangleright$ Multi-scale spatial dependency aggregation
9: \hspace{1cm} $X_{ST} \leftarrow \text{Transformer-Temporal} (X_S)$  \hspace{1cm} $\triangleright$ temporal feature extraction
end for
11: $\hat{Y} \leftarrow \text{Conv} (\text{Conv}(X_{ST}))$  \hspace{1cm} $\triangleright$ Prediction Module
12: return $\hat{Y}$

3.3 TMS Model

In the paper, we defined the traffic graph with sensors as nodes. The spatial dependency of this traffic graph is more complex, not only in the connection relationship between sensor nodes but more importantly in the hidden spatial dependency caused by the area function (e.g., commercial and residential areas are used for shopping and living respectively), road attributes (e.g., speed limit information, number of lanes, road type, etc.), etc. Therefore, when modeling the spatial dependencies, it should also be considered that there will be stronger dependencies between sensor nodes that are in the same area. Also, sensor nodes on roads with the same road attributes will have similar characteristics even if they are physically distant. This requires modeling from different spatial scales to capture a more comprehensive spatial dependency.

The previous models modeled spatial dependence on one scale only. Some models redefine the existence of edges between sensor nodes by a distance threshold, where an edge is considered to exist if the distance between two points is less than this threshold, and the opposite if it is greater, as shown in Fig. 5(a). This approach has a certain subjective factor and ignores the fact that similar road attributes can make nodes that are far apart also have similar features. This method of fixed distance threshold makes the feature extraction very limited and obviously cannot extract spatial features accurately and comprehensively. In addition, as shown in Fig. 5(b), some models use diffusion to gradually capture information from nodes at a distance. That is, the outermost yellowish sensor node passes the information to the innermost yellow sensor node first and then to the A sensor later. This indirect transmission method generates large errors that make feature capture inaccurate. In order to reduce the error, some models choose to treat all sensor nodes as similar nodes for feature extraction in spatial feature extraction. As shown in Fig. 5(c), i.e., the features of all the yellowish sensor nodes are taken into account when the spatial features of sensor
node $A$ are extracted. This improves the accuracy to a certain extent, but this "non-differentiated" feature extraction method will calculate too much useless information. Moreover, the difference in area functions will lead to heterogeneity among areas, and if this heterogeneity is not distinguished, it will lead to inadequate feature extraction and thus cannot model spatial dependencies comprehensively.

Fig. 5 Spatial structure extraction method; taking the extraction of features of sensor node $A$ as an example, (a) indicates that only the sensor nodes identified by the threshold value that has logical connectivity are focused on, i.e., the information transfer is done only between the connected yellow sensors; (b) indicates that the information transfer is done gradually, with the outermost yellowish sensor node passing the information to the innermost yellow sensor node first and then to sensor $A$ gradually afterward; (c) indicates that the feature information of all nodes is passed directly to the $A$ sensor node; (d) indicates that the spatial dependencies at different levels are captured at subscales

To solve this problem, the TMS module is proposed to model spatial dependencies at different spatial scales. TMS is targeted to capture spatial dependencies at different levels, including dynamic spatial dependency feature extraction at three spatial scales, node level, road level, and area level, and static spatial structure feature extraction. As shown in Fig. 5(d), when extracting the features of the red sensor node $A$, the nodes marked with yellow indicate the spatial dependency extraction range at the node level, the nodes marked with pink indicate the extraction range of static spatial structure features, the nodes marked with blue indicate the spatial dependency extraction range at the area level, and the nodes marked with green indicate the spatial dependency extraction range at the road level. Thus, the spatial dependencies at different levels can be modeled in a targeted and refined manner.

Specifically, for the node level, it means that each sensor node has its unique traffic characteristics and does not need to aggregate the characteristics of other sensor nodes. For the road level, it means that nodes with the same properties of surrounding buildings and roads will have similar traffic characteristics. For example, road sections with the same speed limit or the same number of lanes will have similar speed features. This needs to be modeled from a global perspective, as road sections that are far apart may also have the same attributes. For the area level, this means that each sensor node is located within a regional context, and each area has its own unique architectural and functional characteristics. For example, it will have its features in transportation hubs or remote areas. This does not require feature extraction from a global perspective but requires circling the area range where the node is located according to certain rules in order to extract the regional features of the target node.
within that area range. Therefore, to perform accurate and comprehensive spatial dependency modeling, we need to consider multiple scales in order to adequately model dynamic spatial dependencies. In addition, the static spatial structure is also an important part of the composition of spatial features. We will follow the way of defining the logical connection relationship by distance threshold in the previous model, where two sensor nodes are considered connected when the distance between them is less than the threshold, and the opposite when it is greater. This logical connection relation reflects the static spatial structure. The multi-scale spatially dependent aggregation process of TMS is represented as follows:

\[ X_S = \text{GSA}(S_{node}, S_{area}, S_{road}, S_{static}) \]  

where, \( S_{node} \), which equals the original input features, denotes node-level spatial dependencies, \( S_{area} \) denotes area-level spatial dependencies, \( S_{road} \) denotes road-level spatial dependencies, and \( S_{static} \) denotes static spatial structure features, GSA indicates gated fusion operation.

Before the historical data \( X_{in} \in \mathbb{R}^{N \times T} \) is input to the TMS, it is expanded in dimensionality by one \( 1 \times 1 \) convolution layer. The expanded data \( X \in \mathbb{R}^{C \times N \times T} \) is then fed into each feature extraction layer to extract spatial features.

\[ X = \text{Conv}(X_{in}) \]  

(1) Area spatial dependency extraction layer

Spatial dependence modeling is performed on the area-level vehicle speed data \( X \) after expanding the features. Taking the sensor node \( i \) as an example, we take the node \( i \) as the center, determine a region range according to a given distance threshold \( K \), and use the set \( R_i \) to represent all other sensor nodes within this region range. The structure of the area spatial dependency extraction layer is shown in Fig.6 and includes a location embedding unit, a regional multi-headed self-attentive unit, and a feedforward neural network unit.
Position embedding unit

The Transformer mechanism cannot take into account the location information of the data in the sequence. For traffic networks, the location relationship between sensors is very important. In the Position embedding unit, we use a learnable spatial location embedding matrix $R_{\text{area}} \in \mathbb{R}^{N \times N}$ to learn the dynamic location relationships between nodes. $R_{\text{area}} \in \mathbb{R}^{N \times N}$ is initialized to an adjacency matrix constructed from the distances between nodes. After the location embedding, the data are represented as:

$$X' = F (X, R_{\text{area}})$$

where, $F$ is a $1 \times 1$ convolutional layer for incorporating dynamic location information into the data.

Area multi-head self-attention unit

In the area of multi-headed self-attentive unit, $h$ attention heads are used to learn different aspects of the features, and later the results of each attention head are aggregated. In each attention head, the spatial feature $X'$ is extracted. $X'_1, \ldots, X'_i, \ldots, X'_N \in X'$ parallel computing, feature extraction process:

Firstly, three potential subspaces are trained for the sequence of $N$ sensor nodes, including query subspace $Q_{\text{area}}$, key subspace $K_{\text{area}}$, and value subspace $V_{\text{area}}$.

$$Q_{\text{area}} = X'W^q_{\text{area}}$$

$$K_{\text{area}} = X'W^k_{\text{area}}$$

$$V_{\text{area}} = X'W^v_{\text{area}}$$

where, $W^q_{\text{area}}$, $W^k_{\text{area}}$ and $W^v_{\text{area}}$ are the weight matrices for $Q_{\text{area}}$, $K_{\text{area}}$ and $V_{\text{area}}$, respectively.
Secondly, the attention fraction between nodes is calculated. Previous models calculate the spatial dependence \( S_{area} \) between two points by simply computing \( Q_{area} \) and \( K_{area} \) through the dot product as follows:

\[
S_{area} = \frac{Q_{area} \cdot (K_{area})^T}{\sqrt{d_k}} \tag{7}
\]

However, in the above equation, the attention score is calculated for all nodes in the traffic network. And for the area level, this generates a large number of useless calculations and does not consider the regions’ heterogeneity. This can lead to inaccurate learned region features. In this paper, the nodes are filtered in the calculation of regional attention scores, and only the attention scores within their regions are calculated, and the filtering process is as follows:

\[
S_{area}^{ij} = \frac{X_{i}' W_{area}^q \cdot (X_{j}' W_{area}^k)^T}{\sqrt{d_k}} + B_{ij} \tag{8}
\]

where, \( X_{i}' W_{area}^q \) denotes the value corresponding to \( X_{i}' \) in the query subspace, \( X_{j}' W_{area}^k \) denotes the value corresponding to \( X_{j}' \) in the key subspace, and \( \sqrt{d_k} \) is added to prevent the gradient from disappearing and the problem of too large input values. \( B_{ij} \) denotes the filtering variable with a value of 0 when the node \( j \) is within the region of the node \( i \), and vice versa, set to negative infinity.

\[
B_{ij} = \begin{cases} 
0, & j \in R_i \\
-\infty, & j \notin R_i 
\end{cases} \tag{9}
\]

Next, by applying the activation function softmax, the obtained attention scores are mapped to the interval [0,1] to ensure that they always add up to 1 via the sequence. Then multiply and add with the corresponding value subspace to obtain the area characteristics.

\[
M_{area} = \sum_{j \in R_i} \text{softmax}(S_{area}^{ij}) V_j \tag{10}
\]

Finally, the output of the cell is stabilized using a Layer Normalization with residual connections.

\[
M'_{area} = \text{Layer Normalization}(X' + M_{area}) \tag{11}
\]

where, \( M'_{area} \in \mathbb{R}^{C \times N \times d_G \times T} \) denotes the data after learning the spatial features of the area.

* Feedforward neural network unit

The feedforward neural network consists of two linear layers with a non-activating function. To prevent gradient disappearance, a layer normalization operation with residual connections is added after feature extraction to stabilize the output, and the extraction process \( S_{area} \) is:

\[
S_{area} = \text{Layer Normalization}(\text{ReLU}(\text{Linear}(M'_{area}))) + M'_{area} \tag{12}
\]
where, Linear denotes the linear layer used to expand and reduce the dimensionality of the data. ReLU is the nonlinear activation used to learn the nonlinear features of the data.

(2) Road spatial dependency extraction layer

Put $X$ into Road spatial dependency extraction layer. For the node $i$, we take the node $i$ as the center and calculate its dynamic correlation with all nodes. The structure of the road spatial dependency extraction layer is shown in Fig. 7.

$$Q_{road} = X^T W^q_{road}$$  \hspace{1cm} (13)

$$K_{road} = X^T W^k_{road}$$  \hspace{1cm} (14)

$$Q_{road} = X^T W^v_{road}$$  \hspace{1cm} (15)

where, $W^q_{road}$, $W^k_{road}$ and $W^v_{road}$ are the weight matrices of $Q_{road}$, $K_{road}$ and $V_{road}$, respectively.

Secondly, the correlation scores between nodes are calculated. The purpose of feature extraction at the road level is to find roads with similar features in the global

![Fig. 7 Road spatial dependency extraction layer](image-url)
context. Since there is a problem with the large number of nodes to do feature extraction at the road level, there will be a large number of useless computations following the common practice of self-attentive mechanism. In order to reduce the computation without losing accuracy, this paper uses sparse self-attentiveness to extract spatial features. The attention fraction is obtained by first calculating the dot product of $\mathbf{Q}_{\text{road}}$ and $\mathbf{K}_{\text{road}}$:

$$S_{ij}^{\text{road}} = \frac{\mathbf{X}_i \mathbf{W}_q^{\text{road}} (\mathbf{X}_j \mathbf{W}_k^{\text{road}})^T}{\sqrt{\mathbf{d}_k}}$$  \hspace{1cm} (16)

where, $S_{ij}^{\text{road}} \in S^{\text{road}}$. Then we select the highest attention scores and multiply them with the corresponding value subspaces, add them to get the node’s eigenvalues:

$$M_{\text{road}} = \sum F_{\text{top} - p}(S^{\text{road}}) \mathbf{V}$$  \hspace{1cm} (17)

where, $F_{\text{top} - p}$ denotes the filter function, which is used to filter out more valuable data from $S^{\text{road}}$. The $F_{\text{top} - p}$ expression is shown in equation 18.

$$F_{\text{top} - p}(S^{\text{road}}) = \begin{cases} S_{ij}^{\text{road}}, & S_{ij}^{\text{road}} \in \text{top} - p(S^{\text{road}}) \\ 0, & \text{others} \end{cases}$$  \hspace{1cm} (18)

where, $\text{top} - p(S^{\text{road}})$ means the first $p$ data in $S^{\text{road}}$ order of numerical size. The output of this unit is then stabilized using a layer normalization layer with residual connections. Represents the data after learning the spatial features of the area.

$$M'_{\text{road}} = \text{Layer Normalization}(\mathbf{X} + M_{\text{road}})$$  \hspace{1cm} (19)

Finally, a feedforward neural network consisting of two linear layers with a non-activation function is used to learn the nonlinear features of the data. To prevent gradient disappearance, a layer normalization operation with residual connections is added after feature extraction to stabilize the output.

$$S_{\text{road}} = \text{Layer Normalization}(\text{Linear}(\text{ReLU}(\text{Linear}(M'_{\text{road}})))) + M'_{\text{road}}$$  \hspace{1cm} (20)

(3) Static spatial feature extraction layer

For the traffic system, the connectivity relationship of the road sections to which the nodes belong directly affects the spatial dependence of the nodes. So the static spatial structure is also very important for the feature extraction of nodes. In this paper, we will follow the way logical connectivity was defined by distance threshold in the previous model. Two sensor nodes are considered to be connected when the distance between them is less than the threshold value, while the opposite is true when it is greater than the threshold value. This logical connectivity is denoted by the symbol $\mathbf{A}$. Then, based on the logical connection relationship $\mathbf{A}$ between the sensor nodes and the predefined traffic graph, the static features $S_{\text{static}}$ of the traffic are extracted using the graph convolution operation to gather information from its neighboring nodes. The process of extracting the vehicle speed data $\mathbf{X}$ after expanding
the features is as follows:

\[ S_{\text{static}} = \sigma \left( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} XW \right) \]  \hspace{1cm} (21)

where, \( \tilde{A} = A + I \) is the adjacency matrix with additional self-connections. \( \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \) is the degree matrix of \( \tilde{A} \), \( W \) is the trainable weights matrix, and \( \sigma \) is the activation function.

(4) Gated selection aggregating layer

The gated selection aggregating layer uses a gating mechanism to fuse dynamic spatially dependent features at multiple spatial scales and static spatial features. \( g \) is computed based on the data to be fused, and the computed \( g \) is then used to selectively process the input data in a weighted manner. The gate \( g \) is expressed as:

\[ g = \text{sigmoid}(f_{\text{node}}(S_{\text{node}}) + f_{\text{area}}(S_{\text{area}}) + f_{\text{road}}(S_{\text{road}}) + f_{\text{static}}(S_{\text{static}})) \]  \hspace{1cm} (22)

where, \( f_{\text{node}}, f_{\text{area}}, f_{\text{road}} \) and \( f_{\text{static}} \) are linear functions that convert \( S_{\text{node}}, S_{\text{area}}, S_{\text{road}} \) and \( S_{\text{static}} \) to one-dimensional vectors, respectively. The described gating mechanism utilizes a conversion gate and a feed gate to indicate how much output is generated through the conversion input and the feed output, respectively. The extracted spatially dependent traffic data is denoted as:

\[ X_S = g(S_{\text{node}}) + g(S_{\text{area}}) + g(S_{\text{road}}) + (1 - g)(S_{\text{static}}) \]  \hspace{1cm} (23)

3.4 TTW Model

For the traffic system, the spatial structure is not fixed, but highly dynamic. There will be different spatial dependencies at different times. We embed different spatial dependencies structures into corresponding time points and later perform temporal-dependent feature extraction for several time series. At this time, we consider temporal dependencies as containing spatio-temporal dependencies. In order to prevent losing relative position information when extracting temporal dependence, the TTW module is proposed. This module selectively extracts temporal dependent features according to the different contributions of traffic characteristics and relative position information of time series to the prediction. TTW includes Traffic TimeWise Location Embedding Layer, Traffic TimeWise Multi-head self-attention layer, Feedforward neural network layer. The structure is shown in Fig.8.
(1) Traffic TimeWise Location Embedding Layer

For temporal dependencies, the input sequences have a clear temporal sequence, which is very important for traffic data prediction. The original transformer structure is unable to learn the position information of the time series. In order to be able to handle the time series problem, some models use absolute Position Embedding to perform position embedding, i.e., each position in the sequence has a fixed position vector in a way to embed the sequence position information, denoted as:

$$PE(t)_i = \begin{cases} \sin(\omega_i t) & i \% 2 = 1 \\ \cos(\omega_i t) & i \% 2 = 0 \end{cases} \quad (24)$$

The word vectors and location vectors are then summed to obtain the final input for each word, but a study demonstrated that after a series of complex operations, there is no relative location information left [35]. So such a location embedding is not meaningful for strict time series. To solve this problem, instead of using fixed location
codes in extracting spatio-temporal features, this paper injects relative location information into the sequences according to the characteristics of the traffic system. And a trainable parameter representing the relative position is added when calculating the attention score later.

The input $X_S = (X^i_S, \ldots, X^j_S, \ldots, X^{i+T}_S)$ contains $T$ time steps data. There are $2T - 1$ relative position relationships between them. The relative position table is expressed as $RPR = [-T + 1, \cdots, -2, -1, 0, 1, 2, \cdots, T - 1]$, where the relative position of $X^i_S$ and $X^j_S$ is $a_{ij} = j - i \in RPR$. Since traffic data is highly time-fluid data, future time does not affect the direction of the data in previous time steps, so we set all future values in $RPR$ to -1. After that, the corresponding weight matrices are generated separately for $T+1$ kinds of relative position relations, which $a_{ij}$ corresponds to the weight vector $W_{i-j}$, for the subsequent calculation of self-attention. The relative position weight matrix is shown in Fig. 9.

![Fig. 9 The weight corresponding to the relative position](image)

**Traffic TimeWise Multi-head self-attention layer**

In the multi-headed self-attentive layer, $h$ attention heads are used to learn different aspects of the features and later the results of each attention head are aggregated. In each attention head, temporal dependent features are extracted for the input data $X_S \in \mathbb{R}^{C \times T \times N}$. $(X^i_S, \ldots, X^j_S, \ldots, X^{i+T}_S) \in X_S$ are computed in parallel.

Firstly, train the query subspace $Q_T$, key subspace $K_T$, and value subspace $V_T$ for $X_S$.

$$Q_T = X_S W^T_q$$  \hspace{1cm} (25)  \\  
$$K_T = X_S W^T_k$$  \hspace{1cm} (26)  \\  
$$V_T = X_S W^T_v$$  \hspace{1cm} (27)  

where, $W^T_q$, $W^T_k$ and $W^T_v$ are the learnable weight matrices of $Q_T$, $K_T$ and $V_T$ respectively.

Secondly, the dependence between nodes is calculated. The previous model calculates the spatio-temporal dependence $S^T_{ij}$ between two points, which is only obtained
from \( Q_i = X_i^q W_q^T \) and \( K_j = X_j^k W_k^T \) after dot product calculation, which \( \sqrt{d_k} \) is to prevent the dot product from being too large to stabilize the output.

\[
S_{ij}^T = \text{softmax}(\frac{X_i^q W_q^T \cdot (X_j^k W_k^T)^T}{\sqrt{d_k}})
\] (28)

However, in equation 28, the relative position factor between nodes is not considered, which will lead to inaccurate learned features. Different timesteps will have different impacts on the current time. For example, the weight of the impact of the data an hour ago, a day ago, and a week ago on the data at the current moment is different. To distinguish these dependencies, the relative position weights between nodes are added to the attention scores:

\[
S_{ij}^T = \text{softmax}(\frac{(X_i^q W_q^T \cdot (X_j^k W_k^T)^T) W_{i-j}}{\sqrt{d_k}})
\] (29)

Then multiply it with the corresponding value subspace to get the temporal dependence:

\[
M_T = \sum S_{ij}^T V_j
\] (30)

Finally, the output of the unit is stabilized using a layer normalization layer with residual connections:

\[
M_T' = X_S + M_T
\] (31)

(3) Feedforward neural network layer

A feedforward neural network is used to learn nonlinear features. The feedforward neural network consists of two linear layers with a non-activation function. In order to prevent the gradient from disappearing, a layer normalization layer with residual connections is added after the feature extraction to stabilize the output. The extraction process is as follows:

\[
X_{ST} = \text{Layer Normalization}(\text{Linear}(\text{ReLU}(\text{Linear}(M_T')) + M_T'))
\] (32)

3.5 Prediction Module

The prediction module consists of two classical convolutional layers. The first one is used to reduce the dimensionality of the time step, and the second one is used to reduce the dimensionality of the feature dimension. The final traffic data for the next \( T_r \) time steps are obtained, and the dimensionality reduction process is expressed as:

\[
\hat{Y} = \text{Conv}(\text{Conv}(X_{ST}))
\] (33)

Mean absolute loss is adopted to train the model, which \( \hat{Y} \) denotes the predicted value, \( Y \) denotes the true value.

\[
L_{pre}(\hat{Y}, Y) = \|\hat{Y} - Y\|_1
\] (34)

22
4 EXPERIMENT

In this paper, experiments are conducted to evaluate MSS-STT on three public transportation network datasets, PeMSD7(M), PeMSD7(L), and PeMS-BAY. Experiment 1 verifies the overall performance and computational cost of this model on the PeMSD7(M) and PeMS-BAY datasets, which are highly used in the field. In Experiment 2, the baseline model STTN is considered a "simple model" on the PeMSD7(M) dataset, and ablation experiments are designed to verify the effectiveness of the TMS and TTW modules. Experiment 3 was conducted on the PeMS-BAY dataset with a long time span to verify the applicability of the model in different seasons and times. Experiment 4 analyzes and compares the effects of the choice of regional threshold $K$ on the model prediction accuracy in different network size datasets PeMSD7(M) and PeMSD7(L).

4.1 Dataset and preprocessing

PEMSD7(M), PEMSD7(L), and PEMS-BAY contain data on speed data collected from the Caltrans Performance Measurement System (PeMS) and aggregated within 5 minutes. PEMSD7(M) and PEMSD7(L) are published by Yu et al [11]. PEMSD7(M) includes traffic data from 228 sensor stations on the California freeway system for the time period May to June 2012 weekdays. PEMSD7(L) includes traffic data from 1026 sensor stations on the California freeway system for the time period of weekdays from May to June 2012. PEMS-BAY, published by Li et al [3], includes traffic data from 325 sensor stations on the California freeway system for the time period January to May 2017. The details of the datasets are shown in the Table 2.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Node</th>
<th>TimeSteps</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEMSD7(M)</td>
<td>228</td>
<td>12672</td>
<td>2012.5-2012.6</td>
</tr>
<tr>
<td>PEMSD7(L)</td>
<td>1026</td>
<td>12672</td>
<td>2012.5-2012.6</td>
</tr>
<tr>
<td>PEMS-BAY</td>
<td>325</td>
<td>52116</td>
<td>2017.1-2017.5</td>
</tr>
</tbody>
</table>

4.2 Experiment Setting and Evaluation Metrics

(1) Experimental Setting

For a fair comparison with the baseline model, this paper splits the dataset in the same way. All velocity readings are aggregated into a 5-minute window, and we use the one-hour historical velocity data to predict the future one-hour velocity data, i.e., $T=12$. Experiments were conducted in the Tensorflow framework using Intel(R) Xeon(R) Gold 5117, 2.00-GHz CPU, GeForce GTX 2080-Ti 11G GPU, and Adam optimizers were trained on the proposed model with an average absolute error loss of 70 Epochs and a Batch Size of 32. The initial learning rate was 0.01 and decreased at a rate of 0.5 for every twenty Epochs.

(2) Evaluation Metric
In this paper, we use mean absolute error (MAE), root mean square error (RMSE), and mean absolute percentage error (MAPE) to measure the performance of the model.

\[
MAE = \frac{1}{m} \sum_{i=1}^{m} |Y_i - \hat{Y}_i|
\]  

\[
RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (Y_i - \hat{Y}_i)^2}
\]  

\[
MAPE = \frac{100\%}{m} \sum_{i=1}^{m} \frac{\hat{Y}_i - Y_i}{Y_i}
\]

where, \(m\) is the number of batch samples.

4.3 Baselines

MSS-STT is compared with the following models:

1. HA: a simple time-series forecasting model that makes predictions based on the average of historical data, without considering other factors and complex patterns;
2. ARIMA [15]: an autoregressive integrated moving average model with a Kalman filter;
3. FNN: Feed Forward Neural Network;
4. FC-LSTM: Fully connected Long Short-term Memory (LSTM) network;
5. DCRNN [3]: Diffusion Convolution Recurrent Neural Network, which formulates the graph convolution using a diffusion process and integrates the graph convolution into the encoder-decoder module;
6. STGCN [11]: capturing spatial and temporal correlations through spatio-temporal graph convolutional networks;
7. GraphWaveNet [5]: Graph WaveNet augments the GCN using a combination of adaptive adjacency matrices and uses one-dimensional convolution to learn temporal correlation;
8. STSGCN [12]: simultaneous modeling of local spatio-temporal correlations is achieved;
9. STTN [8]: using a spatio-temporal Transformer to model spatio-temporal dependencies;

4.4 Experiment 1: MSS-STT on PEMSD7(M), PEMS-BAY dataset versus baseline model

To validate the overall effectiveness of MSS-STT, this paper uses the publicly available codes of STGCN [11], DCRNN [3], and STTN [8] to train on the PEMSD7(M) and PEMS-BAY datasets, as well as code replication on Graph WaveNet [5] and STSGCN and training on the appealed two datasets. All experimental results are the average results obtained from five independent experiments with the same hyperparameters in the TensorFlow framework. The data comparison of the experimental results is shown in Table 3, and the visualized histograms are shown in Fig. 10, 11 and 12, which
show that the model metrics in this paper reach the first or second best position in the short-term 15mins, 30mins prediction, and long-term 60mins prediction.

Table 3 Comparison of the results of MSS-STT with other models on PEMSD7(M), PEMS-BAY dataset

<table>
<thead>
<tr>
<th>DataSet</th>
<th>Time</th>
<th>Metrics</th>
<th>HA</th>
<th>ARIMA</th>
<th>FNN</th>
<th>FC-LSTM</th>
<th>DCRNN</th>
<th>STGCN</th>
<th>Graph WaveNet</th>
<th>STSGCN</th>
<th>STTN</th>
<th>MSS-STT(ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEMSD7(M)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15mins</td>
<td>MAE</td>
<td>4.01</td>
<td>5.56</td>
<td>2.74</td>
<td>3.57</td>
<td>2.37</td>
<td>2.27</td>
<td>2.15</td>
<td>2.11</td>
<td>2.14</td>
<td>2.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMSE</td>
<td>7.2</td>
<td>9.01</td>
<td>4.75</td>
<td>6.2</td>
<td>4.21</td>
<td>4.04</td>
<td>4.09</td>
<td>4.05</td>
<td>4.04</td>
<td>3.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAPE</td>
<td>10.61</td>
<td>12.92</td>
<td>6.38</td>
<td>8.6</td>
<td>5.54</td>
<td>5.26</td>
<td>4.97</td>
<td>5.04</td>
<td>5.05</td>
<td>4.71</td>
</tr>
<tr>
<td></td>
<td>30mins</td>
<td>MAE</td>
<td>-</td>
<td>5.8</td>
<td>4.02</td>
<td>3.94</td>
<td>3.33</td>
<td>3.05</td>
<td>2.83</td>
<td>2.71</td>
<td>2.69</td>
<td>2.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMSE</td>
<td>-</td>
<td>9.11</td>
<td>6.98</td>
<td>7.03</td>
<td>5.99</td>
<td>5.76</td>
<td>5.5</td>
<td>5.4</td>
<td>5.37</td>
<td>4.91</td>
</tr>
<tr>
<td></td>
<td>60mins</td>
<td>MAE</td>
<td>-</td>
<td>6.25</td>
<td>5.04</td>
<td>4.16</td>
<td>3.86</td>
<td>3.85</td>
<td>3.19</td>
<td>3.24</td>
<td>3.3</td>
<td>2.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMSE</td>
<td>-</td>
<td>9.4</td>
<td>8.58</td>
<td>7.5</td>
<td>7.18</td>
<td>7.2</td>
<td>6.34</td>
<td>6.29</td>
<td>6.31</td>
<td>5.67</td>
</tr>
<tr>
<td>PEMS-BAY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15mins</td>
<td>MAE</td>
<td>2.88</td>
<td>1.62</td>
<td>2.2</td>
<td>2.05</td>
<td>1.38</td>
<td>1.39</td>
<td>1.3</td>
<td>1.91</td>
<td>1.35</td>
<td>1.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMSE</td>
<td>5.59</td>
<td>3.3</td>
<td>4.43</td>
<td>4.19</td>
<td>2.96</td>
<td>2.92</td>
<td>2.72</td>
<td>3.69</td>
<td>2.89</td>
<td>2.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAPE</td>
<td>6.8</td>
<td>3.53</td>
<td>5.19</td>
<td>4.83</td>
<td>2.91</td>
<td>3</td>
<td><strong>2.74</strong></td>
<td>4.16</td>
<td>2.9</td>
<td>2.75</td>
</tr>
<tr>
<td></td>
<td>30mins</td>
<td>MAE</td>
<td>-</td>
<td>2.33</td>
<td>2.3</td>
<td>2.2</td>
<td>1.74</td>
<td>1.84</td>
<td>1.64</td>
<td>1.95</td>
<td>1.67</td>
<td>1.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMSE</td>
<td>-</td>
<td>4.76</td>
<td>4.64</td>
<td>4.55</td>
<td>3.97</td>
<td>4.12</td>
<td>3.65</td>
<td>3.83</td>
<td>3.77</td>
<td>3.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAPE</td>
<td>-</td>
<td>5.47</td>
<td>5.45</td>
<td>5.22</td>
<td>3.94</td>
<td>4.2</td>
<td>3.69</td>
<td>4.53</td>
<td>3.79</td>
<td>3.55</td>
</tr>
<tr>
<td></td>
<td>60mins</td>
<td>MAE</td>
<td>-</td>
<td>3.38</td>
<td>2.46</td>
<td>2.37</td>
<td>2.07</td>
<td>2.46</td>
<td>1.95</td>
<td>2.1</td>
<td>1.95</td>
<td>1.79</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMSE</td>
<td>-</td>
<td>6.5</td>
<td>4.95</td>
<td>4.96</td>
<td>4.79</td>
<td>5.3</td>
<td>4.62</td>
<td>4.51</td>
<td>4.54</td>
<td>4.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAPE</td>
<td>-</td>
<td>8.3</td>
<td>5.91</td>
<td>5.71</td>
<td>4.9</td>
<td>5.58</td>
<td>4.51</td>
<td>5.29</td>
<td>4.6</td>
<td>4.35</td>
</tr>
</tbody>
</table>

Fig. 10 Histogram of evaluation metrics of MSS-STT and baseline models DCRNN, STGCN, Graph WaveNet, STSGCN, STTN predicting velocity data after 15mins
Fig. 11  Histogram of evaluation metrics of MSS-STT and baseline models DCRNN, STGCN, Graph WaveNet, STSGCN, STTN predicting velocity data after 30mins

Fig. 12  Histogram of evaluation metrics of MSS-STT and baseline models DCRNN, STGCN, Graph WaveNet, STSGCN, STTN predicting velocity data after 60mins

Model comparison: Table 3 shows the results of the MSS-STT on the PEMSD7(M), PEMS-BAY dataset compared with 9 baselines. A careful analysis of the results shown in the table leads to the following observations:

The HA model is too simple to capture complex time series patterns; the ARIMA model has high data requirements and needs to satisfy smoothness and linear relationship assumptions; the FNN requires disrupting spatio-temporal data to form a long series, which makes it more difficult to capture temporal and spatial correlations. The FC-LSTM model uses the LSTM module to extract temporal dependencies and spatial dependencies are ignored. As a result, FC-LSTM performed poorly in the evaluation. DCRNN formulates graph convolution with a diffusion process and integrates the graph convolution into the encoder-decoder module, and STGCN captures spatial and temporal correlations through a spatio-temporal graph convolution network; both models take both space and time into account and use graph convolution neural networks to extract features. This approach of extracting spatial dependencies only from static road network structures can have limitations, so the results are still poor. Graph
WaveNet constructs an adaptive matrix to account for changes in influence between nodes and their neighbors, automatically discovering invisible graph structures from the data, a practice that has led to some improvement in prediction accuracy. However, this adaptive matrix is fixed after training, but still cannot be dynamically adjusted with data characteristics. Since these models model static spatial structures and traffic data are highly dynamic, a simple GCN is not sufficient to accurately model spatial dependencies. And these models do not consider spatio-temporal dependence, but calculate spatial and temporal separately, so that the data that have extracted spatial dependence are used in extracting temporal dependence, which will lose part of the original data information. STSGCN tries to combine the spatial-temporal relations for learning and constructs a local spatio-temporal graph, this approach solves the local spatio-temporal dependency problem, but this local approach still loses a large number of spatio-temporal dependencies. Then STTN proposed the spatio-temporal Transformer based on Transformer, which solved the above two problems to some extent and improved the accuracy to a certain degree, but there are still problems of single scale and inadequate feature extraction. And MSS-STT designed the TMS module to model complex spatial structures from different spatial scales to capture spatial dependencies and spatial patterns at different levels, which makes the spatial feature extraction process targeted and reduces a large number of useless calculations. The TTW module is also designed to extract spatio-temporal features based on temporal relative position, which enables the selection of more valuable historical data and improves prediction accuracy to some extent. Compared with models using GNN such as STGCN, our model is more advantageous in long-range prediction (60mins), considering the reason that GNN’s model suffers from the problem of over-smoothing, which makes it difficult to capture spatial correlations over long distances. As shown in Table 3, our model MSS-STT achieved the first or second-best score in each index and improved the prediction accuracy by about 9.27% on average in PEMSD7(M) and about 6.30% on average in PEMS-BAY compared with STTN, which had the best baseline results.

Table 4 Average training speed of MSS-STT with other models on PEMSD7(M) for one Epoch. It can be seen that STGCN is a very efficient method. While DCRNN is more time-consuming because it uses a cyclic structure, which requires sequential computation and updating of model parameters at each time step, which can increase the time cost of computation. In contrast, STTN uses a self-attentive mechanism to model spatial and temporal features, which can deal with the relationship between multiple locations simultaneously, resulting in higher computational parallelism and improved computational efficiency. The increased computational time of MSS-STT compared to STTN is, we believe, due to the TMS module in extracting multi-scale spatial dependencies.

4.5 Experiment 2: Modular ablation experiments of MSS-STT on the PEMSD7(M) dataset

A comprehensive ablation study was conducted to test the effect of each component of the MSS-STT model. The experiments used the baseline model STTN as the base model, i.e., the spatio-temporal dependence was extracted using the ordinary spatio-temporal Transformer mechanism. Keeping all hyperparameters constant, the
Table 4  Average training time (sec/epoch) for PEMSD7(M) obtained by MSS-STT, STTN, GRAPH WAVENET, STGCN and DCRNN, respectively

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average training time (sec/epoch)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSS-STT</td>
</tr>
<tr>
<td>PEMSD7(M)</td>
<td>52</td>
</tr>
</tbody>
</table>

experiment replaces a) the TMS module and b) the TTW module in the MSS-STT model into STTN step by step and observes the metrics on the PEMSD7(M) dataset, and then analyzes the impact and effect of each module on the predictive ability of the model.

As shown in the Table 5, the ablation experiments compare the impact of the components used on the prediction results. Since the spatial dependence in the traffic system cannot be summarized by simple road connection relationships, spatial feature extraction at multiple spatial scales plays an important role in the prediction. In addition, the data in the time series have strict sequential relationships, and not differentiating them would compute a large amount of worthless and introduce more noisy information, which makes the TTW module play an important role.

Table 5  Discussion results of each component of MSS-STT on PEMSD7(M)

<table>
<thead>
<tr>
<th>TMS model</th>
<th>TTW Model</th>
<th>PE MSD7(M)/60mins</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTN</td>
<td>Single spatial feature</td>
<td>Temporal Transformer</td>
</tr>
<tr>
<td>✓</td>
<td>2.96</td>
<td>5.41</td>
</tr>
<tr>
<td>MSS-STT</td>
<td>✓ ✓</td>
<td>2.83</td>
</tr>
</tbody>
</table>

We randomly selected several Epochs during the training process and showed the relative position weight changes in the TTW module as a heat map. It can be seen in Fig.13 that the color is getting lighter when RPR is -1 and darker when RPR is 1-3, which means that RPR is not contributing much to the task of traffic prediction when it is -1 and more to the task of traffic prediction when RPR is 1-3. This verifies the effectiveness of the TTW module by showing that the current traffic data is not affected by future traffic data and that the current traffic data is more affected by traffic data about three time steps away from itself.
4.6 Experiment 3: Experiment and comparison of MSS-STT on PEMS-BAY dataset with a long time span

In order to verify the applicability environment of the model in this paper, extensive experiments and comparisons were conducted on the PEMS-BAY dataset with a broader time span. Fig. 14 and 15 show the visualization of the prediction results on the PEMS-BAY dataset. Our model is able to correctly predict the trend of traffic data in different weather and different time periods, and is more accurate in terms of prediction details compared to other baseline models.
Fig. 14 Graph of experimental results of the same sensor at different times; (a) indicates the velocity forecast for 5.25-5.26; (b) indicates the velocity forecast for 6.29-6.30.

Fig. 15 Different time peak prediction for the same sensor; (a) indicates the peak hour speed forecast details for 5.25-5.26, (b) indicates the peak hour speed forecast details for 6.29-6.30.
4.7 Experiment 4: MSS-STT experiments on setting the region threshold $K$ on PEMSD7(M) and PEMSD7(L) datasets with different network sizes, respectively

Setting reasonable hyperparameters is helpful for model training. Therefore, we discuss the setting and use of regional thresholds in this paper. The data sets used are PEMSD7(M) and PEMSD7(L), and the evaluation metric is MAE. The experimental results are shown in the Fig. 16.

In the dataset PEMSD7(M), the MAE is 2.99 after the experiment by setting the middle value of all distances to 22000, and then the MAE is 2.89 when adjusted to 10000 and 30000 respectively. The MAE results are 2.91, 2.88, 2.83, 2.85, and 2.91 for 11000, 9000, 8000, 7000, and 6000, respectively, and it can be seen that the optimal result is achieved when 8000 is used.

In PEMSD7(L), a dataset with a larger sensor distribution, the MAE is set to the middle value of 66000 for all distance values in the distance matrix of this dataset, and the MAE is 1.98 after the experiment. Therefore, we believe that a value less than the middle value of 66000 will have better results. The MAE results are 1.90, 1.89, 1.87, 1.83, 1.79, and 1.86 for 30,000, 10,000, 9,000, 8,000, 7,000, and 6,000, respectively, and it can be seen that the optimal result is achieved when 7,000 is used.

The above results show that the selection of the area distance threshold is not greatly affected by the size of the dataset network, and the optimal value is achieved in the small range of 7000-8000, both in the smaller network with 228 nodes and in the larger network with 1026 nodes.
5 Conclusion

In this paper, we propose a refined traffic prediction method based on a multi-spatial scale spatio-temporal Transformer. The model includes a multi-scale spatially dependent feature aggregation module, a Traffic TimeWise feature extraction module as well as a prediction module. In TMS, we use several specific spatial Transformer networks as well as graph convolutional neural networks to extract dynamic hidden spatially dependent features at the node level, area level, and road level as well as static spatial structure features, respectively. Second, we use a gating mechanism to fuse data features at each level, which extracts complex traffic spatial dependencies from multiple scales and makes the prediction more accurate and reasonable than other models. In TTW, we use the Transformer network to selectively extract more valuable temporal dependencies based on the characteristics of the traffic data to make the prediction more accurate. The experimental results on real-world datasets demonstrate the superior performance of the proposed MSS-STT, especially in long-term prediction. However, the inclusion of the TMS module increases some time costs while improving the prediction accuracy. In the future, we will improve the TMS module to increase the computational speed. In addition, due to the limited number of samples in the dataset, we believe that the amount of training data is a key element in improving prediction accuracy. In the future, we will consider applying self-supervision within the model to improve the model prediction capability.

6 Declarations

We thank anonymous reviewers for valuable suggestions.

- **Funding** This work was supported in part by “The Double-First-Rate Special Fund for Construction of China University of Mining and Technology, No. 2018ZZCX14.” The funder had no role in study design, data collection and preparation of the manuscript.

- **Conflict of interest/Competing interests** (check journal-specific guidelines for which heading to use) The authors declare no potential conflicts of interest with respect to the research, authorship, and publication of this article.

- **Ethics approval** Not applicable.

- **Consent to participate** Not applicable.

- **Consent for publication** Not applicable.

- **Availability of data and materials** Not applicable.

- **Code availability** https://github.com/ZHANGCUMT1/MSS-STT.

- **Authors’ contributions** YZ and LZ conceived the prediction method, implemented the experiments, conducted the experimental result analysis, and wrote the paper; BL and ZL gathered data and performed experiments. LZ, BL, ZL and XZ revised the paper. All authors have read and approved the final paper.
References


914–921 (2020)


