Article

STN-GCN: Spatial and Temporal Normalization Graph Convolutional Neural Networks for Traffic Flow Forecasting

Chunzhi Wang 1, Lu Wang 1, Siwei Wei 2, Yun Sun 2, Bowen Liu 3 and Lingyu Yan 1,∗

1 School of Computer Science, Hubei University of Technology, Wuhan 430000, China; chunzhiwang@hbut.edu.cn (C.W.); 102101023@hbut.edu.cn (L.W.)
2 School of Computer and Artificial Intelligence, Wuhan University of Technology, Wuhan 430000, China; waosfengw@hotmail.com (S.W.); sunyun@hbut.edu.cn (Y.S.)
3 School of Civil Engineering, Architecture and the Environment, Hubei University of Technology, Wuhan 430000, China; 2110631123@hbut.edu.cn
∗ Correspondence: yanlingyu@hbut.edu.cn

Abstract: In recent years, traffic forecasting has gradually become a core component of smart cities. Due to the complex spatial-temporal correlation of traffic data, traffic flow prediction is highly challenging. Existing studies are mainly focused on graphical modeling of fixed road structures. However, this fixed graphical structure cannot accurately capture the relationship between different roads, affecting the accuracy of long-term traffic flow prediction. In order to address this problem, this paper proposes a modeling framework STN-GCN for spatial-temporal normalized graphical convolutional neural networks. In terms of temporal dependence, spatial-temporal normalization was used to divide the data into high-frequency and low-frequency parts, allowing the model to extract more distinct features. In addition, fine data input to the temporal convolutional network (TCN) was used in this module to conduct more detailed temporal feature extraction so as to ensure the accuracy of long-term sequence extraction. In addition, the transformer module was added to the model, which captured the real-time state of traffic flow by extracting spatial dependencies and dynamically establishing spatial correlations through a self-attention mechanism. During the training process, a curriculum learning (CL) method was adopted, which provided optimized target sequences. Learning from easier targets can help avoid getting trapped in local minima and yields better generalization performance to more accurately approximate global minima. As shown by experimental results the model performed well on two real-world public transportation datasets, METR-LA and PEMS-BAY.

Keywords: spatial-temporal correlation; graph neural network; spatial-temporal normalization; transformer; self-attention mechanism

1. Introduction

With the rapid development of society, urban traffic conditions have become particularly important. At the same time, spatial-temporal characteristic traffic flow forecasting has attracted close attention from various industries. The complexity of traffic networks lies in the forecast of future traffic conditions (e.g., speed, density, and flow) based on historical data. Accurate forecasting of traffic status can be applied to spatial-temporal correlation technologies, making traffic flow forecasting challenging. This forecasting is based on the data collected by sensors distributed in different locations applied to practical problems, such as estimation of the time of future travel and navigation of future traffic routes.

In recent years, graph neural networks have developed rapidly, and spatial-temporal graph modeling has gradually received more attention. It models the interdependence between nodes, constructing a dynamic spatial-temporal network graph that can represent the relationships between nodes. Spatial-temporal graph modeling has multiple applications.
in solving complex system problems. For example, Li et al. investigated the forecasting of traffic speed [1] and Yao et al. predicted taxi demand [2].

At present, traffic flow forecasting is mainly classified into statistical methods models: traditional machine learning models, deep learning models, etc. In general, statistical methods are usually simply used in traffic forecasting to extend the forecast range and improve quasi-time series models; for example, the historical mean model (HA) [3], which uses least squares to dynamically estimate parameters and can make relatively accurate forecasting of changes in traffic status. Apart from that, the vector autoregressive model (VAR) [4] and autoregressive integrated moving average model (ARIMA) [5] are adopted to predict traffic flow from historical temporal data to forecast future traffic flow, but the models require relatively high smoothness of test data. The traditional machine learning models for forecasting, k-nearest neighbor [6] and support vector regression (SVR) [7], are linear and unsuitable for handling fluctuating traffic flow data (e.g., severe weather and holidays), and the forecasting accuracy tends to be low. Among deep learning models, the proposed long short-term memory (LSTM) [8] artificial neural network solves the problem of gradient disappearance and gradient explosion in processing time series by traditional recurrent convolutional networks, and its forecasting effect on traffic data is good and far exceeds that of traditional statistical and machine learning models. However, the model considers the problem of temporal dependence and ignores the spatial dependence of traffic data.

Deep learning models can be deterministic. Recurrent neural network (RNN) models have gained attention for their strong ability to handle series data. However, it still suffers from the problem of gradient disappearance, making it difficult to capture long-term data features, and the training time of the model is long. In order to overcome these problems, convolution was introduced to extract the temporal features of the data [9,10]. Recently Cai, Zheng, et al. [11,12] proposed to incorporate the self-attention mechanism into the forecasting models. In the field of spatial feature extraction, early research is focused on the application of convolutional neural networks (CNN). Given that CNN can only operate in two dimensions, they cannot reflect the topology of traffic networks. For this reason, graph neural networks (GNN) [13–15] have become a popular choice for extracting spatial features from multiple nodes in traffic networks. At present, spatial-temporal traffic forecasting models with more advanced comprehensive performance can not only aggregate the information of neighboring nodes when processing data, but also extract the feature weights of the data to construct adaptive graphs. For example, STFGNN [16] uses train data to create dynamic graphs based on dynamic time warping (DTW) distances. Graph WaveNet [17] adopts a learning node embedding method and constructs adaptive graphs in training. Nonetheless, all of these models build the graph structure during the training phase and do not dynamically adapt to the predicted data. Indeed, the interrelationships among the data nodes may change over time, even at different times of the same day.

In order to overcome the previously mentioned difficulties, inspired by the recent successful experience of applying a transformer in dealing with spatial-temporal correlation modeling, this study introduces a spatial-temporal normalized graph convolutional neural network model (STN-GCN) to collaboratively predict traffic flows at each location on the traffic network. In addition, spatial-temporal normalization is introduced to process the stationarity of data, and a spatial transformer module is added to handle spatial correlations in transportation networks. This paper summarizes the challenges encountered in traffic flow forecasting and proposes relevant solutions to improve forecasting accuracy and efficiency. Its contributions are as follows:

1. This study proposes a spatial-temporal normalized graph convolutional neural network model. The model combines a time convolutional network (TCN) with spatial-temporal feature normalization in the time feature extraction, which can effectively remove noise and more fully extract the temporal feature;
2. This study incorporates a transformer into the deep learning model and uses the spatial-transformer module to extract spatial features from the input data. The model connects the spatial-temporal features extracted after k spatial-temporal modules with residuals. The stacked data features are skip-connection through a fully connected layer to finally output the predicted values;

3. The curriculum learning method is added to the training. Training is conducted in groups. It starts with simple samples and gradually accumulates up to the whole training sample, allowing achieving the best results in training more easily. The results of experiments on two real-world datasets show that the proposed model has an improvement in performance.

2. Background and Related Work

2.1. Graph Neural Networks

A graph neural network model is a specialized deep learning model developed for processing graph data, with an ability to effectively handle the complex spatial-temporal data relationships present in traffic flow forecasting. Roughly speaking, models can be categorized into three distinct groups based on their underlying characteristics and mechanisms:

- Based on graph structure feature extraction [18–20], researchers began to combine graph theory and neural networks to propose a traffic flow forecasting model according to graph structure feature extraction. The graph attention networks (GAT) [21] model proposes a new graph neural network model, the graph attention network, which enables the network to adaptively assign different weights to different nodes through the introduction of an attention mechanism. In comparison to GCN, GAT can more effectively deal with sparse graph data, but it has slightly higher computational complexity. These models mainly use the adjacency matrix of the graph to represent the topology of the road network in order to extract the spatial relationships between different regions;

- Models based on graph convolutional neural networks [22–24] started to apply the ideas of convolutional neural networks (CNN) to graph neural networks and proposed traffic flow forecasting models based on graph convolutional neural networks (GCN). These models can adaptively learn the spatial dependencies between different regions and incorporate them into the forecasting model. However, this method has some problems with computational complexity and does not work well for sparse graphs;

- Models based on spatial-temporal graph [25,26] started to incorporate the temporal dimension into graph neural networks and proposed traffic flow forecasting models based on spatial-temporal graphs. With better predictive capabilities, these models can consider both spatial and temporal-series relationships between different regions.

2.2. Temporal Dependence

As described in [26,27], recurrent neural networks (RNN) are limited by gradient explosion, gradient disappearance, and sequence length uncertainty when modeling temporal dependencies, gated recurrent units (GRU) [28] are designed to alleviate these problems and perform long-term dependent traffic prediction. The incorporation of Graph WaveNet with inflated convolution has been shown to enhance the perceptual field and decrease the number of hidden layers in neural network models. However, this approach encounters limitations when dealing with longer data sequences due to linear increases in the number of hidden layers as a function of sequence length. Furthermore, the ability of the model to effectively capture long-range dependencies between sequence components is severely hampered as the path length between them grows longer. To summarize, because different lengths of input sequences require distinct model designs, it is unfeasible to determine an optimal length. Fortunately, recent advancements in deep learning techniques allow complex dynamics to be treated as a unified entity without the need for additional inputs. In practical applications, multiple time series can be categorized into four groups based on their level of spatial and temporal activation: low-frequency local influences, low-frequency
global influences, high-frequency local influences, and high-frequency global influences. The refined classification of the data into the convolutional layer to take temporal features presents better results than the previous methods.

2.3. Spatial Correlation

Currently, the most common form of graphical neural network uses an adjacency matrix with only structural connectivity information. However, some studies have attempted to incorporate more spatial structural information into traffic prediction. For example, Li et al. [1] suggested trimming edges based on the geospatial distance between nodes to reflect the distance information of the traffic network. As shown by DDP-GCN [29], combining much structural information (e.g., distance, heading direction and joint angle) can enhance the predictive power of the model. However, these methods rely on fixed static information, such as the distance between node pairs, the speed limit of a road segment, and the route angle of two nodes. In spatial-temporal prediction, learning adaptive graphs in the training phase can further improve the performance of the model. Graph WaveNet constructed an adaptive adjacency matrix by multiplying the self-learning node embeddings. However, all these current approaches essentially define the graph before the validation and testing phases. Given that the trends of spatial-temporal data may face changes in daily trends and other unexpected situations during the testing period, a method is needed to adapt the input data for both the training and testing phases. By contrast, the transformer [30] achieves efficient sequence learning through a highly parallel self-attention mechanism that can adaptively capture long-range time-varying correlations from input sequences with different lengths via a single layer.

3. Preparation

For the task of traffic flow prediction, firstly, the definition and construction of the model were described. This paper emphasizes the concept and structure of the STN-GCN model, which was proposed through a combination of spatial-temporal normalized data types and the transformer mechanism.

3.1. Definition of the Traffic Road Network Graph

A traffic road network graph can be defined as a graph $G = (V, E)$, where $V$ represents the set of nodes $V = \{V_1, V_2, \cdots V_n\}$, and it represent a set of connected edges between sensors. The graph consists of $n$ intersection nodes and edges connecting each pair of nodes. The adaptive adjacency matrix $A = A_{ij} \in R^{N \times N}$ is generated from the individual node relationships, where the nodes $V_i, V_j \in V$ are connected by an edge $(V_i, V_j) \in E$. The following Figure 1 is a graph constructed according to the structural relationship of each sensor in the road network, where (a) represents graph $G$ and (b) represents the adjacency matrix. In the Figure 1a, $v$ represents each intersecting node and $e$ represents the connection between each node.

![Figure 1](image-url)
3.2. Feature Matrix

The traffic information observed on the graph is represented as a graph signal $X \in \mathbb{R}^{N \times Q}$, where $Q$ represents the type of each node feature (e.g., speed, flow), and the value of $Q$ in this paper is taken as 1, and the feature is speed. The traffic prediction problem is eventually transformed into learning a function $f(\cdot)$. It is assumed that $X(t)$ denotes the graph signal observed at moment $t$. Mapping $T'$ signals on historical graphs to future $T$ graph signals is to predict the traffic state at a future time period based on the traffic information of the past time period, and the prediction process can be expressed by Equation (1):

$$[X_{t+1}, \ldots, X_{t+T}] = f(G_i(X_{t-n}, \ldots, X_{t-1}, X_t))$$ \hspace{1cm} (1)

From a graph-theoretic standpoint, multivariate time series data can be conceptualized as individual nodes on a graph, Figure 1b the relationships between these nodes can be captured through the use of a graph adjacency matrix. In many cases, the graph adjacency matrix is not directly provided in multivariate time series data, but can be learned through models designed for this purpose.

4. Methodology

4.1. General Model Framework

Figure 2 shows the model framework proposed in this paper, which consists of a temporal convolution module (STT-Block) and a spatial-transformer module (STT-Block) for the extraction of spatial-temporal features and then an output module. The model first takes the traffic flow data as input and then converts the data type through a linear layer. The converted data flows into the spatial-temporal normalization module (ST-Norm) for normalization and is passed to two parallel gated temporal convolution modules (TCN). To circumvent the issue of vanishing gradients, a residual connection is established between the output of the graph convolutional module and the input of the temporal convolutional module through the addition of residual values. This connection ensures that information from earlier layers within the network is preserved as the model progresses toward the output layer. The extracted temporal features are passed to the spatial-transformer module to extract spatial features. Firstly, the model propagates the extracted temporal features to the spatial transformer module for spatial feature extraction. Subsequently, the spatial-temporal features obtained from the spatiotemporal feature extraction module (ST-Module) are passed on to the next layer to continue feature extraction. It is worth noting that the outputs obtained from $K$ ST-Modules are seamlessly integrated into the output layer through skip connections.

![Figure 2. General model diagram.](image)
4.2. Temporal Extraction Module (STT-BLOCK)

Spatial and temporal normalization modules are added before the data flow into the spatial-temporal convolution module to polish the high-frequency element (temporal) and low-frequency element (spatial) of the original data, respectively [31]. The modules “low-frequency” and “high-frequency” are utilized to characterize the degree to which extrinsic disturbances affect a system from a temporal standpoint, while “global” and “local” denote the scale of impact from a spatial perspective. The modules “low-frequency” and “high-frequency” describe the degree of temporal variability in a signal, where low-frequency elements indicate sluggish changes that remain relatively constant over extended periods, whereas high-frequency elements denote sudden, sharp fluctuations. “Global” refers to the impact on all-time series being similar, while “local” indicates that the effect may be restricted to a single time series or may differ among multiple ones. By comparing pairs of time series that share the same global component over time, the module is capable of isolating the local component of a single time series. An arbitrary time series can be decomposed into four components: local high and low frequencies and global high and low frequencies, respectively:

\[
M = M_{ll}M_{lh}M_{gl}M_{gh}
\] (2)

Temporal normalization (TN) is aimed to optimize and filter the high-frequency elements (both global and local) from the mixed signal. This study introduces two symbols to generalize the high-frequency element and the low-frequency element, respectively, which can be denoted as:

\[
M_{\text{high}} = M_{lh}M_{gh}
\] (3)

\[
M_{\text{low}} = M_{ll}M_{gl}
\] (4)

The applicability of T-Norm in the context of time series problems is predicated on the logical conjecture that individual low-frequency elements roughly approximate a constant value across a given period. It is possible to apply T-Norm on a time series without additional complementary features representing the frequencies. This property is quite suited for many realistic problems in which specific frequencies are unavailable. \(E(M_{\text{low}})\) and \(\sigma^2(M_{\text{low}})\) denote the mean and standard deviation of the low-frequency elements of the local data, respectively. \(\varphi\) represents a small numerical constant for maintaining numerical stability. \(M\) represents the input time series. \(\delta\) represents the period of time during which the low-frequency elements remain approximately constant. When calculating the mean, by setting \(\delta\) equal to the input time step, the calculation steps become relatively straightforward and more intuitive. \(\gamma_{\text{high}}\) and \(\beta_{\text{high}}\) refer to the mean and standard deviation (positive and negative) of the effect of high frequencies on the time series, respectively. T-Norm style represents the low-frequency element and the content represents the high-frequency element, which can be expressed as:

\[
E(M_{\text{low}}) = \frac{1}{\delta} \sum_{t'=1}^{\delta} M_{t-\delta+1}
\] (5)

\[
\sigma^2(M_{\text{low}}) = \frac{1}{\delta} \sum_{t'=1}^{\delta} M_{t-\delta+1} - E(M_{t})^2
\] (6)

\[
M_{\text{high}} = \frac{M - E(M_{\text{low}})}{\sigma(M_{\text{low}}) + \varphi} \gamma_{\text{high}} + \beta_{\text{high}}
\] (7)

The objective of spatial normalization (S-Norm) is to optimize the local elements comprising both high-frequency and low-frequency elements. The feasibility of S-Norm
depends on the conjecture that the impact of the global element on the all-time series is uniform in nature. Here local and global high and low-frequency elements are introduced:

\[ M_{\text{global}} = M_{\text{gh}} M_{\text{gl}} \]  
\[ M_{\text{local}} = M_{\text{lh}} M_{\text{ll}} \]  

where \( E(M_{\text{global}}) \) and \( \sigma(M_{\text{global}}) \) can be deduced from the equations described above through replacement of approximate values of the four latent variables into Equation (10):

\[ M_{\text{local}} = M - E(M_{\text{global}}) \sigma(M_{\text{global}}) + \gamma_{\text{local}} + \beta_{\text{local}} \]  

The S-Norm corresponds to the T-Norm in the spatial field, where the high-frequency elements are used as local elements and the low-frequency elements conform to the global elements. The model is spatially and temporally distinguishable and can be fitted exclusively to each sample set, especially the long-tailed sample clusters. By extracting the local or high-frequency elements from the primary signal, the rank of the feature space can be increased, meaning that it enables the model to obtain more different features. Though experimental, it is validated that the model can obtain more subtle changes in the data, which is quite fruitful in extracting the time-dependent process of the data.

In the wake of data normalization, as shown in Figure 2b, the dilation causal convolution [32] is used as the temporal convolution layer (TCN) to obtain the temporal dependencies of the nodes. Dilated convolution can be applied to regions larger than its own length by skipping some of the inputs, and an exponential increase of the perceptual field can be achieved by increasing the depth, thereby improving the model’s computational efficiency and reducing the complexity. The gating mechanism [33] is essential for controlling the flow of information between layers in recurrent neural networks and is equally powerful in controlling temporal convolutional networks. Given a time series input \( X_t \in \mathbb{R}^T \) with a single feature, and a convolution kernel \( \omega \in \mathbb{R}^P \), the causal convolution of the expansion can be expressed as:

\[ X_t^{(T)} \ast \eta_\omega = \sum_{\rho=0}^{P} \omega(\rho)X_t^{(T)}(t - d \times \rho) \]  

where \( \ast \eta_\omega \) represents the expanded convolution operation with kernel \( \omega \), \( d \) represents the expansion factor, and the scalar values in parentheses denote the indexes of the vectors. Finally, the input sequence \( X_t \in \mathbb{R}^{N \times T \times C} \) is passed to the gated activation unit through the expanded causal convolution to extract the temporal features of the input sequence as follows:

\[ H_t = \sigma(\Phi_1 \ast \eta X_t^{(T)}) \odot \tanh(\Phi_2 \ast \eta X_t^{(T)}) \]  

where \( \Phi_1 \) and \( \Phi_2 \in \mathbb{R}^{P \times C \times D} \) refer to the kernels of the extended arbitrary convolution, \( \odot \) denotes multiplication by elements, and \( \sigma(\cdot) \) denotes the sigmoid activation function.

4.3. Spatial Extraction Module (ST-BLOCK)

The purpose of the spatial extraction module (spatial-transformer) is to integrate the information between nodes and their neighboring nodes in order to deal with the spatial dependencies present in the node graph. The features of internodal relationships are calculated by means of an aggregation process utilizing the localized information of neighboring nodes, which is done in accordance with both the predefined graph architecture and the learned weighting matrix. The module consists of four parts: spatial location embedding layer, static graph convolution layer, dynamic graph convolution layer, and gating mechanism. Among them, the spatial location embedding layer integrates the spatial location information such
as topology, connectivity, and time step into each node. The static graph convolution layer and the dynamic graph convolution layer explore the spatially dependent smoothness and directed dynamic components, respectively, and finally obtain the graph with spatial structure by fusing these two convolution layers. The gating mechanism fuses the extracted information and inputs it to the next layer for more efficient information processing. The architecture of spatial transformer is shown in Figure 2c.

4.3.1. Spatial Location Embedding Layer

Transformer is a model based on a self-attention mechanism that is different from recursive and convolutional neural networks in terms of local connectivity and parameter sharing. In order to ensure the unique representation of each token in the input sequence and the preservation of distance information, position encoding needs to be added to each token. During this process, the spatial location embedding layer uses $P$ minimal nontrivial feature vectors as the location embedding of nodes and learns the spatial embedding of each node feature through a learnable spatial location embedding layer. It utilizes the graph adjacency matrix for spatial dependency modeling, considers the connectivity and distance between nodes, and takes the dictionary $D$ as the learning of the spatial location embedding to process the input sequence.

4.3.2. Static Graph Convolution Layer

For graph structure data, the graph $G$ is constructed based on the physical connectivity and distance between sensors, so that the arrangement of GNN is fixed, and how to design the location encoding remains a problem to be solved. Therefore, the fixed spatial dependencies determined by the road topology can be explored explicitly through a fixed graph convolution layer. For this purpose, this study adopts the Laplace location encoding approach and computes the Laplace feature vector of the input graph as follows:

$$L = T - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U^T \bigwedge U$$

(13)

where $D$ represents the degree matrix of $G$, $A$ represents the adjacency matrix, $\bigwedge$ represents the eigenvalue matrix, and $U$ represents the eigenvector matrix. The non-normalized matrix $A$ has the potential to disrupt the original feature distribution when it is multiplied with the feature matrix, thereby resulting in unpredictable complications. To address this concern, a normalization procedure is performed on matrix $A$. Firstly, we guarantee that the sum of each row in matrix $A$ equals 1 by multiplying it with the inverse of the degree matrix $D$. Subsequently, we decompose the inverse of $D$ and conduct a multiplication with $A$, yielding a symmetric and normalized matrix. Consequently, $D^{-\frac{1}{2}}$ represents the decomposed degree matrix.

4.3.3. Dynamic Graph Convolution Layer

This study proposes a new dynamic graph convolution layer for training and modeling high-dimensional potential subspaces. The proposed technique involves linearly mapping the input features of each node into an appropriately high-dimensional subspace. Self-attention mechanisms are then employed to effectively capture the dynamically evolving spatial dependencies between all nodes in the projected feature space. Unlike previous approaches, the edge weights calculated based on the predefined road topology used in [34] cannot adequately characterize the dynamic spatial dependencies in the traffic network. Therefore, this study learns multiple linear mappings to model dynamic directional spatial dependencies influenced by various factors in various potential subspaces.

The embedded features $X_S$ for each time step are first projected into a high-dimensional latent subspace. The mapping is implemented using a feed-forward neural network. $X_S$ is first projected to three matrices: query $Q$, key $K$, and value $V$, which can be expressed as:

$$Q = X_SW_Q$$

(14)
where \( W^Q, W^K, \) and \( W^V \) represent the projection matrices acting on \( Q, K, \) and \( V \). \( Q \) and \( K \) have the same dimension \( D_{QK} \), and the experiment sets \( D_{QK} \) equal to the dimension \( D_V \) of \( V \). Then, the self-attention can be written as:

\[
\text{Attention}(X_S) = \text{softmax} \left( \frac{Q^K \sqrt{D_{QK}}} V \right)
\]

(17)

The dot product is used to reduce the computational and storage costs in the computation. Softmax is adopted to normalize the spatial dependence and scale \( \sqrt{D_{QK}} \) is used to prevent saturation due to Softmax functions.

4.3.4. Gating Mechanism for Feature Fusion

The gating mechanism is applied to fuse the spatial features learned from the static and dynamic graph convolution layers. The gating \( g \) is derived from \( Y_S \) and \( X_G \) of the static and dynamic graph convolution layers:

\[
g = \text{sigmoid}(F_S(\tilde{Y}_S) + F_G(X_G))
\]

(18)

where \( F_S \) and \( F_G \) represent linear projections that transform \( Y'_G \) and \( X_G \) into one-dimensional vectors, respectively.

\[
Y_S = gY'_G + (1 - g)X_G
\]

(19)

Ultimately, the output \( Y_S \) is obtained by weighting \( Y'_G \) and \( X_G \) with the gating \( g \). The output obtained by weighting each spatial-temporal layer is fed to the output layer through a jump connection, and the final predicted value is output via the output layer.

5. Experiments

5.1. Experimental Setups

This experiment was conducted on a computer with Intel(R)Core(TM)i7-11800H @2.30 GHz and NVIDIA GeForce RTX 3060 graphics card. This study used an 8-layer Graph WaveNet model with an expansion factor of 1,2,1,2,1,2,1,2 sequence, a convolutional kernel size of 2, and a batch size set to 16. Moreover, the optimization of the model was performed through the utilization of the Adam optimizer and initialized with a learning rate of 0.001.

5.2. Data Description

This study validated the model using two real-world case public transportation network datasets, METR-LA and PEMS-BAY. Particularly, the METR-LA dataset contained four months of traffic speed statistics recorded by 207 sensors on Los Angeles urban freeways, while the PEMS-BAY dataset contained six months of traffic speed information from 325 sensors in the Bay Area. The adjacency matrix between each node was constructed from the Gaussian distance of the road network, and each sensor was collected at 5-min intervals. These datasets were divided in chronological order, with 70% used for training, 10% for validation, and 20% for testing. Table 1 details the statistical information of the datasets.

<table>
<thead>
<tr>
<th>Data</th>
<th>Nodes</th>
<th>Edges</th>
<th>Time Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>METR-LA</td>
<td>207</td>
<td>1515</td>
<td>34,272</td>
</tr>
<tr>
<td>PEMS-BAY</td>
<td>325</td>
<td>2369</td>
<td>52,116</td>
</tr>
</tbody>
</table>
The present investigation employed the pedagogical technique of curriculum learning (CL) [35] to train a model. CL emulates the human learning process by directing the model, to begin with rudimentary samples and gradually advance to more intricate ones, thus facilitating knowledge acquisition. Extensive research has demonstrated that the adoption of the CL approach enhances the generalization capacity and convergence rate of models in numerous applications [36], including computer vision and natural language processing. Empirical evidence confirms that the implementation of CL in model training leads to notable improvements in performance quality by effectively utilizing instructional materials of varying difficulty levels.

5.3. Evaluation Indicators

Traffic flow prediction tends to be evaluated using the mean absolute error (MAE), root mean square error (RMSE), and mean absolute percentage error (MAPE) for the prediction results. Let \( x = x_1, \ldots, x_n \) be the historical speed value observed by the sensor, \( y = y_1, \ldots, y_n \) denote the predicted speed value, and \( \Omega \) denote the number of observed samples. Then, the three evaluation metrics can be calculated as:

\[
\text{MAE}(x, y) = \frac{1}{\Omega} \sum_{i \in \Omega} |x_i - y_i| 
\]

\[
\text{RMSE}(x, y) = \sqrt{\frac{1}{|\Omega|} \sum_{i \in \Omega} (x_i - y_i)^2} 
\]

\[
\text{MAPE}(x, y) = \frac{1}{|\Omega|} \sum_{i \in \Omega} \frac{|x_i - y_i|}{x_i} 
\]

5.4. Experimental Results

The model proposed in this paper (STN-GCN) was compared with the current mainstream and classical traffic flow prediction models for experiments. Average speed prediction values at three distinct time points (i.e., 15, 30, and 60 min) were generated through testing of datasets METR-LA and PEMS-BAY. Subsequently, the predicted values were evaluated using an appropriate function, and the resulting comparison of prediction performances across the various models is shown in Table 2.

<table>
<thead>
<tr>
<th>Data</th>
<th>Models</th>
<th>15 min</th>
<th>30 min</th>
<th>60 min</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>RMSE</td>
<td>MAPE</td>
<td>MAE</td>
</tr>
<tr>
<td>METR-LA</td>
<td>ARIMA</td>
<td>3.98</td>
<td>8.21</td>
<td>9.60%</td>
</tr>
<tr>
<td></td>
<td>DCRNN</td>
<td>2.77</td>
<td>5.36</td>
<td>7.28%</td>
</tr>
<tr>
<td></td>
<td>ASTGCN</td>
<td>2.70</td>
<td>5.24</td>
<td>6.89%</td>
</tr>
<tr>
<td></td>
<td>STGCN</td>
<td>2.88</td>
<td>5.74</td>
<td>7.62%</td>
</tr>
<tr>
<td></td>
<td>GraphWaveNet</td>
<td>2.85</td>
<td>6.30</td>
<td>6.90%</td>
</tr>
<tr>
<td></td>
<td>STN-GCN</td>
<td>2.69</td>
<td>5.15</td>
<td>6.89%</td>
</tr>
<tr>
<td>PEMS-BAY</td>
<td>ARIMA</td>
<td>1.62</td>
<td>3.30</td>
<td>3.50%</td>
</tr>
<tr>
<td></td>
<td>DCRNN</td>
<td>1.63</td>
<td>2.95</td>
<td>3.01%</td>
</tr>
<tr>
<td></td>
<td>ASTGCN</td>
<td>1.48</td>
<td>3.01</td>
<td>3.02%</td>
</tr>
<tr>
<td></td>
<td>STGCN</td>
<td>1.36</td>
<td>2.96</td>
<td>2.90%</td>
</tr>
<tr>
<td></td>
<td>GraphWaveNet</td>
<td>1.30</td>
<td>2.74</td>
<td>2.80%</td>
</tr>
<tr>
<td></td>
<td>STN-GCN</td>
<td>1.31</td>
<td>2.73</td>
<td>2.74%</td>
</tr>
</tbody>
</table>
As shown by the comparison, the model outperforms all other traditional time-series models. In comparison to other methods, this paper rejects the convolution of traditional fixed-node graphs, learns different node parameters for different data to optimize the model structure, and uses dynamic graph expression capability with a certain magnitude of improvement. It not only significantly outperforms the previous convolution-based method ASTGCN, but also outperforms the recursive-based method DCRNN. Although DCRNN applies a circular convolutional model to extract temporal features, it is not effective in extracting features for longer time series, and the error is different from this model by nearly 5%. The present model has a small improvement over DCRNN in the 15-min range, but a larger improvement is achieved in the 60-min horizon. The gated temporal convolution in this paper solves this problem well. In comparison to ASTGCN, Graph WaveNet, and FC-LSTM, the spatial-transformer module in this paper makes the acquisition of graph information more efficient and the error is correspondingly reduced by about 5%.

This study arbitrarily captures a segment of the predicted value of the time period 60 min prior and plots it against the actual value in Figure 3. As shown by the results, the effect of STN-GCN prediction is closer to the overall direction of the real value, and the error is within the acceptable range.

![Figure 3. Comparison of predicted and actual values in 60 min.](image)

Figure 4 shows a comparison of the performance of this model and the benchmark model on different time prediction tasks, where Figure 4a represents the experimental result on the METR-LA dataset and Figure 4b represents the experimental result on the PEMS-BAY dataset. It can be observed from the graphs that all the metrics vary smoothly and are better than the previous benchmark model.

![Figure 4. Comparison of the changes of different model metrics on different data sets.](image)
6. Conclusions

In conclusion, this paper proposes a temporal normalized graph convolutional neural network (STN-GCN) model for traffic flow prediction. The model incorporates spatial-temporal normalization to preprocess the input data before passing it into the temporal convolution layer, facilitating the extraction of temporal features from urban road traffic data, particularly for medium and long-term prediction. Moreover, the transformer architecture dynamically captures spatial dependencies across multiple scales, which further enhances the extraction of spatial features. Additionally, the integration of curriculum learning techniques improves the experimental results. In terms of prediction performance at different time intervals, the proposed model exhibits smoother transitions and consistently outperforms other models. Furthermore, it surpasses the benchmark model in terms of its ability to predict traffic flow over medium and long durations. As part of our future work, we plan to explore the integration of STN-GCN with other deep learning models to uncover latent structured features within the input data.

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

Funding: This research was funded by the National Natural Science Foundation of China under Grant, grant number 61772180 and the APC was funded by 61772180.

Data Availability Statement: The authors approve that data used to support the findings of this study are included in the article. The data used in the article can be downloaded from the following link. GitHub-benchoi93/PeMS-BAY-2022. https://download.csdn.net/download/weixin_41990278/86781398?utm_source=bbsseo (accessed on 20 June 2023).

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

References


