International Journal of Geo-Information

MDPI

Article

SASTGCN: A Self-Adaptive Spatio-Temporal Graph Convolutional Network for Traffic Prediction

Wei Li [1], Xi Zhan [2], Xin Liu [1], Lei Zhang [3], Yu Pan [4] and Zhisong Pan [1,*]

[1] Command and Control Engineering College, Army Engineering University of PLA, Nanjing 210007, China; liwei@aeu.edu.cn (W.L.); liuxin@aeu.edu.cn (X.L.)
[2] Nanjing Research Institute of Electronic Engineering, Nanjing 210007, China; 1300062806@pku.edu.cn
[3] Academy of Military Science, Beijing 100091, China; zhanglei@aeu.edu.cn
[4] College of Systems Engineering, National University of Defense Technology, Changsha 410073, China; panyu0511@nudt.edu.cn
* Correspondence: panzhisong@aeu.edu.cn

Abstract: Traffic prediction plays a significant part in creating intelligent cities such as traffic management, urban computing, and public safety. Nevertheless, the complex spatio-temporal linkages and dynamically shifting patterns make it somewhat challenging. Existing mainstream traffic prediction approaches heavily rely on graph convolutional networks and sequence prediction methods to extract complicated spatio-temporal patterns statically. However, they neglect to account for dynamic underlying correlations and thus fail to produce satisfactory prediction results. Therefore, we propose a novel Self-Adaptive Spatio-Temporal Graph Convolutional Network (SASTGCN) for traffic prediction. A self-adaptive calibrator, a spatio-temporal feature extractor, and a predictor comprise the bulk of the framework. To extract the distribution bias of the input in the self-adaptive calibrator, we employ a self-supervisor made of an encoder–decoder structure. The concatenation of the bias and the original characteristics are provided as input to the spatio-temporal feature extractor, which leverages a transformer and graph convolution structures to learn the spatio-temporal pattern, and then applies a predictor to produce the final prediction. Extensive trials on two public traffic prediction datasets (METR-LA and PEMS-BAY) demonstrate that SASTGCN surpasses the most recent techniques in several metrics.

Keywords: traffic prediction; spatio-temporal sequence; self-adaptive; graph convolutional network

Citation: Li, W.; Zhan, X.; Liu, X.; Zhang, L.; Pan, Y.; Pan, Z. SASTGCN: A Self-Adaptive Spatio-Temporal Graph Convolutional Network for Traffic Prediction. *ISPRS Int. J. Geo-Inf.* **2023**, *12*, 346. https://doi.org/10.3390/ijgi12080346

Academic Editor: Wolfgang Kainz

Received: 28 June 2023
Revised: 5 August 2023
Accepted: 15 August 2023
Published: 18 August 2023

1. Introduction

Traffic prediction issues have become a crucial element of the Intelligent Traffic System (ITS) in recent years [1]. As seen in Figure 1, it has been extensively utilized in numerous disciplines, including traffic speed forecasting [2], flow prediction [3], trip time estimation [4], bike-sharing allocation [5], and taxi order response (pick-up and drop-off) [6]. Reliable forecasts are increasingly essential to develop sensible travel and transportation strategies.

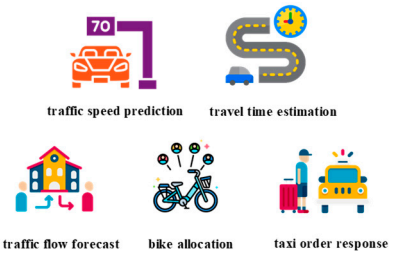

Figure 1. Some applications of traffic prediction.

As a typical spatio-temporal forecasting task, traffic prediction is quite intractable, mostly as a result of the intricate spatio-temporal dependencies and dynamic changes in

spatio-temporal distribution. On the one hand, a high correlation exists between the spatial and temporal characteristics of traffic data, which conduces that they ought to be captured simultaneously. On the other hand, the temporal distribution of the data is susceptible to all kinds of external influences. For instance, road construction, new subway stations, and sudden weather changes are among the factors that could impact statistical values, such as mean and variance in traffic data.

Conventional techniques commonly employ recurrent neural networks, including long short-term memory (LSTM) [7], gated recurrent unit (GRU) [8], and their derivatives to capture temporal dependencies in traffic data. Furthermore, convolutional neural networks (CNN) [9] are utilized to model spatial correlations between regions in grid-based traffic data, whereas graph neural networks (GNN) [10] are applied to graph-based traffic data. More recently, transformer-based architectures have been introduced to facilitate long-term traffic prediction.

However, these approaches neglect the fact that the statistical information of traffic data distributions (such as mean and variance) varies over time, thus leading to unsatisfactory prediction performance and poor generalization ability. Passalis et al. [11] introduced a deep adaptive neural network that is capable of dynamically learning temporal distribution shifts, thereby allowing the model to comprehensively predict both future sequences and anticipated mean and variance. Arik et al. [12] proposed a self-adaptive forecasting model that can adaptively encode the evolving distributions. Nonetheless, they failed to tackle the issue of concurrent distribution shifts across multiple correlated time series, and the dynamic shifts in the temporal distribution of traffic data cannot yet be modeled.

To fix the previously mentioned issues, we developed a hybrid deep learning framework named Self-Adaptive Spatio-Temporal Graph Convolutional Network (SASTGCN). The framework mainly comprises three components: a self-adaptive calibrator, a spatio-temporal feature extractor, and a predictor. In the self-adaptive calibrator, we exploit a self-supervisor composed of an encoder–decoder module to derive the distribution bias of the input, and both the encoder and decoder consist of two layers of recurrent units. Taking the concatenation of the bias and the original features as input, the spatio-temporal feature extractor utilizes graph convolution structures as well as a transformer [13] layer to learn the spatio-temporal pattern. Moreover, a predictor, which consists of a fully connected layer, is applied to produce the final forecast. As far as our knowledge extends, this is the primary attempt to analyze the temporal distribution shift in spatio-temporal prediction issues using an autoencoder. The principal contribution of this work can be succinctly summarized in the following manner:

- We construct a novel self-adaptive calibrator, which can obtain the temporal distribution of traffic data. The calibrator exploits a self-supervised encoder–decoder structure, where the encoder and decoder are both constructed using recurrent layers to better capture the temporal dynamic properties.
- We propose a spatio-temporal feature extractor to discover both spatial and temporal dependencies simultaneously by stacking ST blocks with residual connections. The ST blocks consist of graph convolution layers for spatial correlations and a transformer layer for temporal characteristics.
- Our model surpasses the state-of-the-art methods, as evidenced by a comprehensive range of experimental outcomes on two real-world benchmark datasets: METR-LA and PEMS-BAY.

2. Related Work

As a quintessential problem in spatio-temporal sequence prediction, traffic prediction plays a fundamental role in the advancement of smart cities. Consequently, it has attracted considerable attention from scholars and practitioners alike, leading to its rapid development as a research discipline. There are now two broad groups of traffic forecast methods: statistical techniques and data-driven techniques. Owing to a shortage of transportation data and processing resources, statistical methods dominated the early stages

of traffic prediction. Representative techniques in this category include support vector regression (SVR) [14], auto-regressive integrated moving average (ARIMA) [15], logistic regression (LR) [16], localized extended Kalman filter (L-EKF) [17], and gradient boosting decision tree (GBDT) [18]. They overlook the long-term temporal relations and consider the spatio-temporal sequence individually, which is far from satisfactory.

As data acquisition and deep learning methods advance swiftly [19], data-driven approaches become mainstream. Compared to statistical models, data-driven models are better equipped to capture the highly non-linear complex features in large-scale spatio-temporal data. Several neural network (NN) approaches, including artificial NN [20] and deep belief networks (DBNs) [21], are adopted in traffic prediction during the preliminary stage. To mine the temporal correlations, researchers introduced recurrent neural networks such as LSTM into traffic prediction and showed promising performance [22]. Liu et al. introduced a prediction framework that utilizes LSTM with feature partitioning and feature selection, which successfully enhanced prediction performance by incorporating feature engineering techniques [23]. The traffic state in a region is probably impacted by its surrounding regions as well as distant regions (for instance, there exists a subway between two regions or the functionality of two regions is relevant). To fully employ the node-to-node spatial correlations, a novel model called ST-ResNet [24] was put forth by Zhang et al. in which the entire city is segmented into a regular grid map and the traffic data is projected as a series of images. By this means, a convolution neural network is exploited to obtain the spatial correlations. To further capture the temporal dependencies, Yao et al. introduced a model DMVST-Net [25] that combines CNN, graph embeddings, and LSTM to extract the spatio-temporal feature.

Nevertheless, the approaches mentioned above are only suitable for regular traffic, while, in reality, the majority of traffic data are irregular non-Euclidean data, and projecting into the grid-like sequence would be quite harmful to the prediction accuracy. The GCN is a kind of network structure that is suitable for dealing with non-Euclidean data. Yu et al. originally applied graph convolutional networks to traffic prediction areas. They proposed a model ST-GCN that captures the spatial and temporal features using gated 1D convolution and graph convolution, respectively. Owing to replacing the recurrent layers with 1D convolution, the computational cost is reduced considerably, and thus the model became faster. However, the adjacency matrix in ST-GCN is pre-defined and static, while the spatial dependency between regions changes over time. Graph WaveNet [26] was invented by Wu et al.; in it, a novel flexible dependency matrix is developed and learned through node embedding. Combining a stacked dilated 1D convolution, Graph WaveNet can handle very long sequences as well as dynamic spatial correlations. A single graph could only represent one perspective of the relationship among different regions, while the correlation between regions is manifold. Li et al. proposed a spatio-temporal fusion graph neural network (STFGNN) [27] that, by combining different spatial and temporal graphs, could efficiently discover hidden spatio-temporal dependencies. As reported in [28], Cai et al. combined GCN with a transformer to model the spatio-temporal correlations and periodicity present in traffic data. However, none of these methods have successfully addressed the issue of data distribution shifts in spatio-temporal sequences.

3. Preliminaries

3.1. Traffic Prediction

Traffic prediction is a typical spatio-temporal sequence prediction task. We formulate the road network for traffic as a graph $G =< V, E, A >$, where V denotes a finite set of nodes ($|V| = N$, N represents the total count of nodes, and each node corresponds to a specific region on the road network); E contains a set of edges between the nodes described above and A denotes the spatial adjacency matrix representing the similarity of nodes. If $V_i, V_j \in V$ and $(V_i, V_j) \in E$, then A_{ij} equals 1; otherwise, A_{ij} takes the value of 0. A common traffic prediction problem is shown in Figure 2 and can be succinctly described as follows:

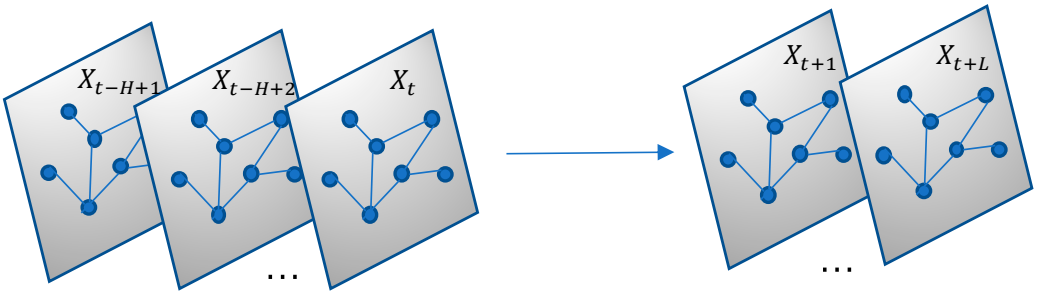

Figure 2. Graph-structured traffic data prediction.

Given the previous H traffic observations, it aims to forecast the traffic observations (speed, demand, flow, and so on) in the next L time steps, which can be defined as follows:

$$X_{t-H+1:t} \xrightarrow{f} X_{t+1:t+L}, \tag{1}$$

where $X_{t-H+1:t} \in R^{H\times N\times d}$ represents the traffic observations from time step $t-H+1$ to t (the value of d corresponds to the dimension of the observation). f denotes the model function to be learned and it estimates the most likely traffic observations $X_{t+1:t+L} \in R^{L\times N\times d}$ in the next L time steps.

3.2. Graph Convolutional Network

The standard convolution for regular grids is not suitable to handle graph-structured data, i.e., graphs, whereas the graph convolutional network has demonstrated its superior performance in handling them. Graph convolution is a process employed to derive characteristics via aggregating information from neighborhood nodes on the graph, which is similar to what normal convolution operations do on images. Nodes in the graph constantly change their state under the influence of nearby and distant points until reaching a final equilibrium, with closer neighbors exerting stronger influence. Spectral-based and spatial-based approaches are two common categories of GCN. The former approaches apply convolutional filters in the spectral domain with graph Fourier transforms [29,30] and the latter combines the representation of the node with that of its neighbors to obtain a novel representation for the node [31,32]. Based on the theory mentioned above, given a defined graph $G =< V, E, A >$, a common paradigm of graph convolutional operation can be defined as follows:

$$S^{(i+1)} = f(S^{(i)}, A), \tag{2}$$

where $S^{(i)}$ and $S^{(i+1)}$ is the graph signal (feature) in the i and $i+1$ layer, respectively, and $S^{(0)}$ is the initial input of the graph. A denotes the adjacency matrix of the graph, and f is the aggregation function. Specifically, Equation (2) can be written as follows:

$$\Theta_{*\mathrm{G}} S^{(i+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}S^{(i)}\Theta), \tag{3}$$

where "$_{*\mathrm{G}}$" is the convolution operator, Θ denotes the kernel, σ represents a non-linear transformation function, $\hat{A} = A + I_N$ denotes the self-looping adjacency matrix, and $\hat{D}$ indicates the diagonal degree matrix of $\hat{A}$.

4. Methodology

This section commences with an overview of the comprehensive architecture of our proposed model, SASTGCN. Subsequently, each component of SASTGCN is formally described. Finally, a detailed explanation is provided on how to optimize the algorithm.

4.1. Overall Architecture

Previous graph-based approaches have primarily concentrated on exploring the dynamic changes in spatial correlation by designing various adjacency matrices. Nevertheless,

these approaches have overlooked the fact that the temporal distribution of data also changes over time. For instance, the construction of new traffic infrastructure, unexpected weather changes, or new public policies implemented by the government can exert an influence on the mobility patterns of individuals, thus instigating modifications in the temporal distribution of traffic data. Therefore, it is crucial to devise a novel approach that takes into account spatial as well as temporal variations in the data to acquire a more holistic comprehension of the underlying patterns and trends. Figure 3 demonstrates the architecture of SASTGCN, which is constituted of three parts: (1) the self-adaptive calibrator; (2) the spatio-temporal feature extractor; and (3) the predictor. The self-adaptive calibrator constitutes a self-supervised encoder–decoder framework, with each of its encoder and decoder modules composed of two recurrent layers, which are adept at capturing temporal relations. The spatio-temporal feature extractor takes as input the merged bias and original features, and it utilizes graph convolution structures as well as a transformer layer to learn the spatio-temporal pattern. To resize the feature and produce the final output, a predictor consisting of a fully connected layer is additionally employed. The detailed implementation of each component is expounded in the ensuing sub-sections.

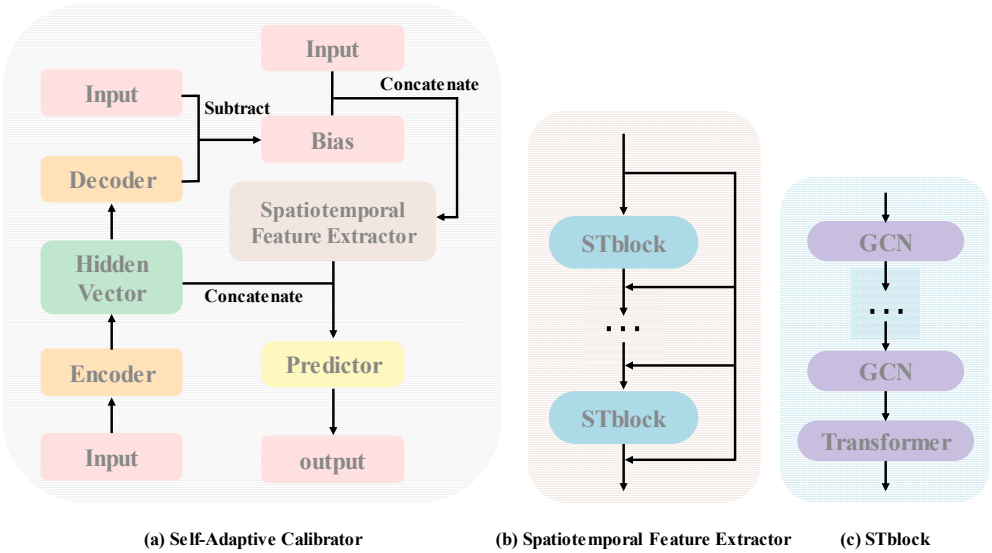

Figure 3. The overall architecture of our proposed SASTGCN.

4.2. Self-Adaptive Calibrator

In the self-adaptive calibrator part, we employ a self-supervised encoder–decoder structure to capture the temporal distribution of data. The encoder and decoder are each comprised of two recurrent layers. Given the input traffic observations $X_{in} \in R^{H \times N \times d}$, we first utilize an encoder to model it and obtain the hidden feature X_{hidden}, which could be represented as follows:

$$X_{hidden} = LSTM_2(LSTM_1(X_{in})), \tag{4}$$

where $LSTM(\cdot)$ denotes a long short-term memory layer as depicted in Appendix A, and the second LSTM layer ingests the hidden states of the first LSTM layer as input. The decoder utilizes the same network structure as the encoder to convert the hidden representation X_{hidden} into the encoder–decoder output X_{out1}, which can be formulated as below:

$$X_{out1} = LSTM_4(LSTM_3(X_{hidden})). \tag{5}$$

Then the bias of the output and original input is calculated as $X_{bias} = X_{in} - X_{out1}$, which denotes the dynamic changes in the spatio-temporal pattern. Finally, we generate the new feature X_{new} by concatenating the model input X_{in} and the bias X_{bias}, and it will be sent to spatio-temporal feature extraction for further processing. The equation is shown below:

$$X_{new} = X_{in} \oplus (X_{in} - X_{out1}), \tag{6}$$

where $\oplus$ denotes the concatenation operation in dimension and $X_{new} \in R^{2H \times N \times d}$. By combining the raw data X_{in} with the biases resulting from distribution shifts X_{bias}, X_{new} is capable of effectively representing the dynamic changes in traffic data distribution.

We have utilized a joint training framework, which integrates the autoencoder loss and prediction loss described in Section 4.5, to concurrently optimize both components. This approach combines the reconstruction ability of the autoencoder with the prediction capability of the model. As a result, this joint training framework leads to enhanced performance.

4.3. Spatio-Temporal Feature Extractor

Taking the preprocessed feature X_{new} as input, the spatio-temporal feature extractor is utilized to dig out the sophisticated spatio-temporal correlations hidden behind the data. The basic component of this module is a block, which is composed of two graph convolutional layers used to explore spatial characteristics, and it is followed by a transformer structure designed to extract temporal dependencies. To enhance the information-capturing ability and global view of the spatio-temporal feature extractor, we sequentially arrange a series of ST block and employ residual connections to mitigate gradient vanishing while simultaneously accelerating the training procedure of the model (we set the number of ST block $p = 3$). The whole process of this module is shown in Algorithm 1, and the implementation details are described below.

Algorithm 1: Implementation of Spatio-temporal Feature Extractor (STFE)

Input: Processed feature $X_{new} \in R^{2H \times N \times d}$, ST block number p, GCN layer number q.
Output: spatio-temporal feature $X_{st} \in R^{2H \times N \times d}$.

```
Set X_temp = X_new, X_st = X_new //initialized as the feature extractor input
for i = 1, 2, ..., p do          //stack p ST blocks with residual connection
    for j = 1, 2, ..., q do      //used to implement a single ST block
        X_st = GCN(X_st)         //a graph convolutional layer
    end for
    X_st = Transformer(X_st)     //a transformer structure to learn temporal patterns
    X_st = X_temp + X_st         //implement the residual connection function
end for
return X_out
```

Figure 3c shows the inner structure of one single spatio-temporal block ST block. It takes the calibrated features $X \in R^{2H \times N \times d}$ as the input. Firstly, the feature is sent to a two-layer GCN structure, and the adjacent matrix is calculated via SVD decomposing. We reshape the input 3D tensor $X \in R^{2H \times N \times d}$ as a 2D tensor X_a shaped $(2H \cdot d \times N)$. To obtain the inner correlations between distinct regions, we apply the single value decomposition algorithm (SVD) to perform matrix factorization on X_a, resulting in the transformation of X_a into two new matrices:

$$X_a = X_t (X_r)^T, \tag{7}$$

where X_r and X_t respectively symbolize the region-wise and the time-wise matrices. The matrix $X_r \in R^{N \times \lambda}$ comprehensively contains spatial correlations amongst all regions, where λ indicates the region dimension. To compute the resemblance between the i-th and j-th region, we utilize a Gaussian kernel-based method to estimate their similarity. The similarity between stations can be reckoned as follows:

$$A_{ij} = exp(-\frac{||X_s(i) - X_s(j)||^2}{\varepsilon^2}), \tag{8}$$

where ε denotes the standard deviation and A_{ij} is used as the adjacency matrix. On top of this, we construct two graph convolutional layers GCN_1 and GCN_2, which share the same

structure except for the parameter. After obtaining spatial features $X_{sp} = GCN_2(GCN_1(X))$, we employ a transformer structure to extract the temporal pattern from X_{sp} and eventually get the spatio-temporal feature X_{st} as shown below:

$$X_{st} = Transformer(X_{sp}), \tag{9}$$

where Transformer$(\cdot)$ denotes a transformer structure, and its implemented details are shown in Appendix B due to space constraints.

Figure 3b depicts the residual connection of ST blocks; for the l-th layer, the output of the layer $X_{st}(l+1)$ is formulated as follows:

$$X_{st}(l+1) = STblock(X_{st}(l)) + X_{st}(0), \tag{10}$$

where $STblock(\cdot)$ is the single ST block aforementioned and $X_{st}(0)$ is equal to X_{st}.

4.4. Predictor

The predictor is comprised of a linear layer and a reshape operation. This part takes the concatenation of spatio-temporal feature X_{st} from the STFE module in Section 4.3 and the hidden feature X_{hidden} of the calibrator in Section 4.2 as the input $X_{final} \in R^{2H \times N \times d}$:

$$X_{final} = X_{st} \oplus X_{hidden}, \tag{11}$$

which is then delivered to a fully connected linear layer and transformed accordingly to conform to the prescribed prediction shape. The process could be represented as follows:

$$X_{prediction} = \delta(WX_{final} + b), \tag{12}$$

where $\delta(\cdot)$ is an activation function and W and b are parameters to be trained ($X_{prediction} \in R^{L \times N \times d}$, here L denotes the prediction time steps).

4.5. Training

The whole procedure of our proposed model SASTGCN is depicted in Algorithm 2. It should be noted that the self-adaptive calibrator is trained synchronously, that is to say, the entire model constitutes a seamless end-to-end pipeline. The training loss L can be expressed in the following manner:

$$L = L_1 + \mu L_2, \tag{13}$$

where L_1 denotes the loss of model input and prediction, L_2 indicates the calibrator loss (encoder–decoder loss), L_1 and L_2 are MSE (mean squared error) losses, and μ is a hyperparameter that balances the weight of these two losses, which is set to 0.5 in practice.

Algorithm 2: Self-Adaptive Spatio-Temporal Graph Convolutional Network (SASTGCN)

Input: Initial input $X_{in} \in R^{H \times N \times d}$.//sample chosen from preprocessed dataset.

Output: model prediction $X_{out} \in R^{L \times N \times d}$.//corresponding predictions for the input.

1. $X_{hidden} = Encoder(X_{in})$ //use a two-layer LSTM structure to encode the input
2. $X_{out1} = Decoder(X_{hidden})$ //decode the hidden representation via a decoder
3. $X_{new} = X_{in} \oplus (X_{in} - X_{out1})$//join the initial input X_{in} with its deviation from X_{out1}
4. $X_{st} = STFE(X_{new})$ //generate the spatio-temporal feature
5. $X_{final} = X_{st} \oplus X_{hidden}$ //concatenate the feature with the hidden representation
6. $X_{out} = Linear(X_{final})$ //generate the prediction X_{out} and reshape it
7. **return** X_{out}

5. Experiments

We execute comprehensive experiments on two real-world traffic datasets, METR-LA and PEMS-BAY, to substantiate the efficacy of the proposed approach with empirical evidence. In addition to evaluating its performance against other baselines, ablation research has been conducted to confirm the functionality of several modules in SASTGCN. Furthermore, we conduct a rigorous examination of the impact of the hyperparameter on the efficacy of the model.

5.1. Datasets

We verify our model on two publicly available spatio-temporal traffic datasets, METR-LA and PEMS-BAY, released by Li et al. (DCRNN) [2]. METR-LA compiles data collected by 207 sensors over four months to provide statistics on traffic speed along the highways of Los Angeles County, encompassing the period between 1 March 2012, and 30 June 2012. The detailed information of METR-LA dataset is illustrated in Appendix D. The PEMS-BAY has documented an extensive six-month period of traffic speed information, commencing on 1 January in the year of our Lord two thousand and seventeen and culminating on 31 May in the same year, sampled from 325 sensors in the Bay Area. As for the spatial adjacency network, we creatively employ the SVD methods to obtain a dynamic adjacency matrix to better cope with dynamic spatial correlation changes. We aggregate these two datasets into 5 min windows. The dataset statistics details are shown in Table 1.

Table 1. Dataset description.

Dataset	Nodes	Samples	Sampling Interval	Edges	Missing Ratio
METR-LA	207	34,272	5 min	1515	8.109%
PEMS-BAY	325	52,116	5 min	2369	0.003%

5.2. Baselines

We use the following baselines as a comparison. For the sake of fairness, we tune the key hyperparameters to ensure that they have the best performance.

HA: The historic average method is a forecasting technique that assumes that traffic flow follows a seasonal pattern and predicts future traffic by taking the average of past observations. An example of implementing this approach would involve utilizing all recorded data from 5:00 p.m. to 6:00 p.m. on Mondays throughout history as a reference point to forecast traffic speed for the same time frame on the upcoming Monday.

SVR [14]: Support vector regression employs a linear support vector machine to perform regression tasks, which allows for a certain degree of deviation between the actual and predicted values.

FC-LSTM [33]: The FC-LSTM model represents an encoder–decoder architecture that leverages the long-short term memory (LSTM) neural network with a peephole mechanism. Notably, both the encoder and decoder components are composed of two distinct recurrent layers.

DCRNN [2]: Similar to FC-LSTM, it is a diffusion convolutional recurrent neural network that represents a sophisticated encoder–decoder architecture, comprising a recurrent layer that enables the network to effectively process sequential data. Nevertheless, in recurrent layers, the matrix multiplications are replaced with diffusion convolution.

GraphWaveNet [26]: Graph WaveNet introduces a pioneering adaptive adjacency matrix concept, which is incorporated into the graph convolution technique employing 1-D dilated convolutions to learn the dynamic and long-term spatio-temporal correlations.

MTGNN [34]: MTGNN is a spatio-temporal framework that integrates graph learning, graph convolution, and temporal convolutional modules.

SLCNN [35]: SLCNN combines structure learning convolution blocks with a pseudo-three-dimensional convolution module to model the spatio-temporal correlations in traffic speed data.

AutoCTS [36]: AutoCTS employs a combination of micro and macro search spaces to represent potential architectures of ST blocks and connections between them to obtain the optimal forecasting models.

STFGNN [27]: STFGNN fuses several spatial and temporal graphs and applies a gated convolution layer to handle the long-term sequence prediction problem.

MD-GCN [37]: MD-GCN employs a dual graph convolution network operating across multiple temporal scales, which is comprised of a gated temporal convolution and a dual graph convolution module.

5.3. Experimental Setup

We partition the METR-LA and PEMS-BAY datasets into discrete training, validation, and testing sets apportioned at a ratio of 7:1:2. The basic time interval is set to 5 min, and we leverage the observed traffic values spanning throughout 12 time intervals to forecast the ensuing values in the next 3, 6, and 12 time intervals, respectively. We set the aforementioned parameters to align with the ones documented in the literature [1] to establish a fair comparison setting. To assess the effectiveness of the approaches, we employ a triumvirate of commonly accepted metrics, namely the mean absolute error (MAE), the rooted mean squared error (RMSE), and the mean absolute percentage error (MAPE). All three criteria are specified in Appendix C for simulation, and the smaller the numerical value, the better the model performs. All experiments were executed on a 64-bit Ubuntu Server equipped with a 2.40 GHz GPU and a plenty ensemble of 8 NVIDIA Titan GPUs, and the codes were implemented by the PyTorch (https://pytorch.org/, accessed on 8 May 2023) framework. In the realm of neural network-based methodologies, the optimal hyperparameters were selected through a rigorous grid search process that was predicated upon the performance evaluation of the validation set. We optimized the whole model with the Adam optimizer, whose learning rate was set to 0.001. The training epoch was 200 and an early-stopping mechanism was utilized with 10 patient epochs.

5.4. Main Results

Tables 2 and 3 present the primary outcomes of SASTGCN on two real-world datasets. The superior results from the experiment are emphasized in bold. Moreover, ten-fold cross-validation was conducted to calculate the average error for each value, which is then denoted by the $\pm$ symbol to represent the error range. Upon observation, it is evident that our proposed method, SASTGCN, outperforms all others in each of the three evaluation metrics, thereby proving its superiority and applicability. It is noticeable that traditional methods (HA, SVR) are far less effective than deep learning methods. The poor effect of FC-LSTM is likely because it completely ignores the spatial correlations among regions. Additionally, short-term forecasting outperforms long-term forecasting in terms of accuracy and precision. This phenomenon can be attributed to the accumulation of prediction errors in the early stages of long-term forecasting, which subsequently impact the accuracy of later stages. In conclusion, by incorporating a calibrator mechanism and transformer architecture to extract spatio-temporal correlations among regions, our proposed SASTGCN model outperforms all comparative methods in the majority of instances.

Table 2. Prediction efficacy of divergent approaches on the METR-LA dataset.

Model	METR-LA (15 min/30 min/60 min)		
	MAE	**RMSE**	**MAPE (%)**
HA	4.16/4.16/4.16	7.80/7.80/7.80	13.00/13.00/13.00
SVR	3.99/5.05/6.72	8.45/10.87/13.67	9.30/12.10/16.70
FC-LSTM	3.44/3.77/4.37	6.30/7.23/8.69	9.60/10.90/13.20
DCRNN	2.77/3.15/3.60	5.38/6.45/7.59	7.30/8.80/10.50
GraphWaveNet	2.69/3.07/3.53	5.15/6.22/7.37	6.90/8.37/10.01
MTGNN	2.69/3.05/3.49	5.18/6.17/7.23	6.86/8.19/9.87
SLCNN	**2.53**/2.88/3.30	5.18/6.15/7.20	6.70/8.00/9.70

Table 2. *Cont.*

Model	METR-LA (15 min/30 min/60 min)		
	MAE	**RMSE**	**MAPE (%)**
AutoCTS	2.67/3.05/3.47	5.11/6.11/7.14	6.80/8.15/9.81
STFGNN	2.57/2.83/3.18	4.73/5.46/6.40	**6.51**/7.46/8.81
MD-GCN	2.65/2.99/3.43	5.09/6.06/7.15	6.82/8.19/10.04
SASTGCN(ours)	2.63 ± 0.04/**2.86 ± 0.05**/**3.15 ± 0.05**	**4.52 ± 0.04**/**5.50 ± 0.04**/**6.34 ± 0.05**	6.77 ± 0.08/**7.41 ± 0.08**/**8.73 ± 0.09**

Table 3. Prediction efficacy of divergent approaches on the PEMS-BAY dataset.

Model	PEMS-BAY (15 min/30 min/60 min)		
	MAE	**RMSE**	**MAPE (%)**
HA	2.88/2.88/2.88	5.59/5.90/5.59	6.80/6.80/6.80
SVR	1.85/2.48/3.28	3.59/5.18/7.08	3.80/5.50/8.00
FC-LSTM	2.05/2.20/2.37	4.19/4.55/4.96	4.80/5.20/5.70
DCRNN	1.38/1.74/2.07	2.95/3.97/4.74	2.90/3.90/4.90
GraphWaveNet	1.30/1.63/1.95	2.74/3.70/4.52	2.73/3.67/4.63
MTGNN	1.32/1.65/1.94	2.79/3.74/4.49	2.77/3.69/4.53
SLCNN	1.44/1.72/2.03	2.90/3.81/4.53	3.00/3.90/4.80
AutoCTS	1.30/1.61/1.89	2.71/3.62/4.32	2.69/3.55/4.36
STFGNN	1.16/1.39/**1.66**	**2.33**/3.02/3.74	2.41/**3.02**/3.77
MD-GCN	1.32/1.64/1.92	2.81/2.71/4.40	2.77/3.66/4.45
SASTGCN(ours)	**1.15 ± 0.03**/**1.37 ± 0.02**/1.79 ± 0.03	2.45 ± 0.05/**2.64 ± 0.04**/**3.68 ± 0.06**	**2.38 ± 0.03**/3.05 ± 0.03/**3.61 ± 0.04**

5.5. Effect of Each Component

We have undertaken a meticulous ablation study on the PEMS-BAY dataset to ascertain the efficacy of the pivotal components that significantly enhance the overall performance of our model. For the sake of convenience, we name SASTGCN without the different components below:

- w/o HR: SASTGCN without the hidden representation added in the predictor. We utilize the output of STFE to a linear layer to generate the prediction straightly.
- w/o Calibrator: SASTGCN without the whole calibrator part. We sent the initial input to the STFE module to derive the ultimate prediction.
- w/o Transformer: SASTGCN without the transformer part in each ST block. We eliminate the temporal feature module in the STFE module.
- w/o RC: SASTGCN with the residual connection in the STFE module. We stack three ST blocks together in a basic configuration.

We conducted an experiment on the PEMS-BAY dataset, with a forecasting horizon of six timesteps (30 min). The results are shown in Table 4. It is noteworthy that SASTGCN outperformed the variants that lacked a hidden representation in the predictor, which indicates that the hidden representation can preserve information and assist in the prediction. Additionally, SASTGCN performed better than the variants without a calibrator and a transformer, demonstrating the effectiveness of the calibrator and the indispensability of the temporal module. Furthermore, the residual connection was also found to be beneficial. In summary, each of the designed sub-modules mentioned above had a positive impact on improving overall performance.

Table 4. Ablation study on the PEMS-BAY dataset (horizon = 6).

Approaches	w/o HR	w/o Calibrator	w/o Transformer	w/o RC	SASTGCN
MAE	1.47	1.68	1.72	1.53	**1.37**
RMSE	3.24	3.54	4.21	3.02	**2.64**
MAPE (%)	3.83	4.24	5.69	3.87	**3.04**

5.6. Parameter Sensitivity Study

The hyperparameter input horizon has a crucial impact on the model performance, which is essential in SASTGCN. The impacts of different horizon lengths on the model prediction outcomes are shown for the datasets METR-LA in Figure 4. The values of the input horizon length H are 3, 6, 12, 24, 48, and 72; the forecast horizon is set to 6. The RMSE error lowers at first before rising. We can see that the input horizon length of 12 yields the greatest results in the SASTGCN model. The RMSE decreases as length increases because the model has more historical data at its disposal. Nevertheless, performance continues to decline as H increases above 12 points. One possible explanation is that as the sequence length increases, the model becomes much more complicated and the prospect of overfitting presents a considerable concern as it bears the potential to adversely affect its overall performance.

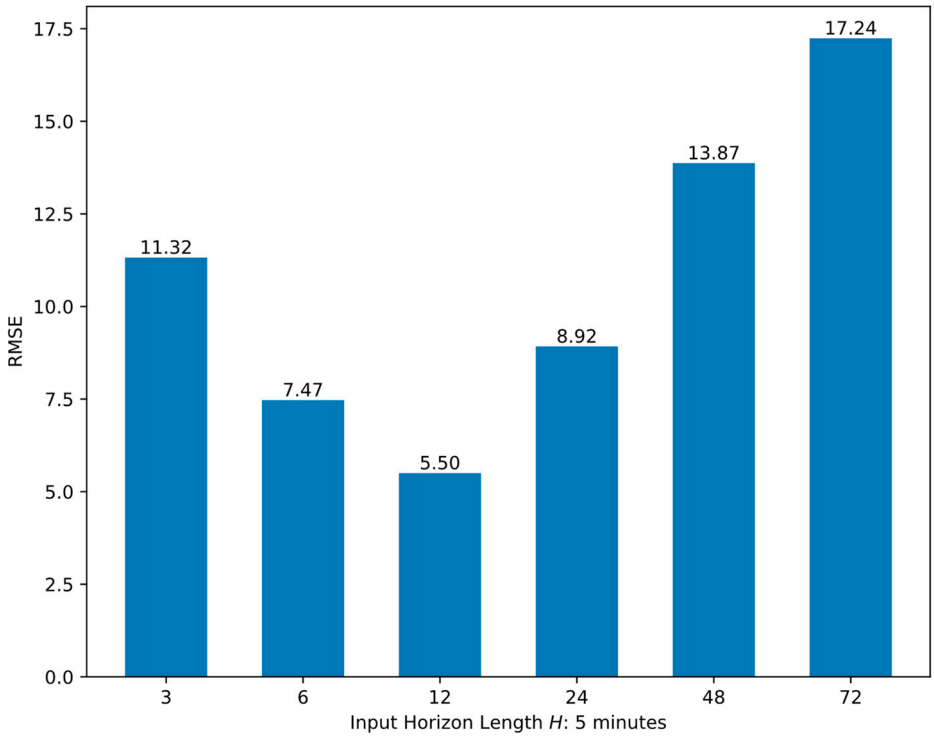

Figure 4. The influence of input horizon length H on the METR-LA dataset.

6. Conclusions

In this paper, we proposed a novel Self-Adaptive Spatio-Temporal Graph Convolutional Network (SASTGCN) for traffic prediction. SASTGCN can effectively capture complex and dynamic spatio-temporal relationships and model the temporal distribution shift by combining a self-adaptive calibrator with graph convolution. To capture the temporal drifts in distribution, we employ a calibrator consisting of an encoder–decoder framework made up of multiple LSTM layers. Furthermore, we stack a series of spatio-temporal modules via residual connections to extract spatio-temporal features from the data. Each spatio-temporal module comprises GCN layers and a transformer encoder, where GCN is utilized to extract spatial correlations among regions, and the transformer is employed to capture temporal characteristics. The SVD method was employed in constructing the adjacency matrix for GCN. Extensive experimental results on two real-world datasets have demonstrated the effectiveness of our proposed model.

For future work, we aim to improve the performance of our model by incorporating external features, such as weather, holidays, and events. It is worth noting that SASTGCN is not only limited to traffic prediction but can also serve as a general framework for spatio-temporal sequence prediction in various domains. As a result, we intend to adapt the

proposed SASTGCN to other prediction scenarios, including weather, energy, agricultural yield, and social media prediction, among others.

Author Contributions: Conceptualization, Xi Zhan; methodology, Wei Li and Xi Zhan; software, Wei Li; Validation, Lei Zhang , Xin Liu and Yu Pan; formal analysis, Wei Li; investigation, Xin Liu; resources, Wei Li; data curation, Lei Zhang and Yu Pan; writing—original draft preparation, Wei Li; writing—review and editing, Wei Li and Xin Liu; visualization, Wei Li; supervision, Zhisong Pan; project administration, Zhisong Pan; funding acquisition, Zhisong Pan 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, grant number 62076251.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here https://github.com/liyaguang/DCRNN, accessed on 8 May 2023.

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

Appendix A

The principles and definitions of Long Short-Term Memory in Section 4.2 are elaborated here in detail.

As a typical representative of recurrent neural networks (RNN), long short-term memory (LSTM) addresses the issue of vanishing gradients present in conventional RNNs. It has been extensively employed in the realm of natural language processing, speech recognition, and other sequence modeling tasks.

The equation of LSTM entails the incorporation of three sophisticated gates and a memory cell to meticulously regulate the information flow throughout the neural network. The input gate, forget gate, and output gate serve as the guardians of information flow, while the memory cell dutifully preserves long-term knowledge. The function of the input gate is to selectively designate information from the present input that merits retention within the memory cell. The forget gate discerns the relevance and necessity of former remembrances within the memory cell to be relinquished. Finally, the output gate is responsible for determining which specific information from the memory cell ought to be transmitted to the output. The equation of LSTM can be expressed formally as described below:

$$
\begin{aligned}
i_t &= \sigma(W_i \cdot [x_t, h_t - 1] + b_i), \\
f_t &= \sigma(W_f \cdot [x_t, h_t - 1] + b_f), \\
o_t &= \sigma(W_o \cdot [x_t, h_t - 1] + b_o), \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tanh(W_C \cdot [x_t, h_t - 1] + b_C), \\
h_t &= o_t \cdot \tanh(C_t),
\end{aligned}
$$

where i_t, f_t, and o_t respectively assume the role of the input gate, forget gate, and output gate at time step t. C_t denotes the cell state and h_t is the hidden state at time step t. x_t represents the input at time step t. W and b are corresponding trainable weight and bias matrices. σ is the sigmoid function, and $\cdot$ denotes element-wise multiplication.

Appendix B

This appendix describes the transformer formulation in Section 4.3.

The transformer, an eloquently conceived neural network paradigm introduced by Google in 2017, remains a stalwart model employed for intricate natural language processing (NLP) endeavors. Compared to traditional convolutional neural networks (CNNs) and recurrent neural networks (RNNs), the transformer has better performance and faster training speed when processing long sequence data. The multi-head self-attention mechanism serves as the fundamental core of the transformer's architecture, which can establish long-range dependencies between different positions, thereby better capturing the information in the sequence. The transformer also includes two modules, the encoder and

decoder, which effectively map the input sequence onto the hidden space and subsequently decode the resulting representation into the output sequence. The detailed equation of the transformer is as follows:

1. Self-Attention Mechanism

In the transformer, the self-attention mechanism is used to calculate the correlation between each word and other words to establish context relationships. Specifically, the representation vector of each word can simultaneously serve as a query vector, key vector, and value vector. The equation is as stated below:

$$Attention(Q, K, V) = soft\max(\frac{QK^T}{\sqrt{d_k}})V,$$

where $Q, K,$ and V are query vector, key vector, and value vector, respectively. d_k is the dimension of K. Through meticulous computation of the similarity between the inquiry vector and the key vector, the weight vector is eventually derived. Subsequently, the weight vector is subjected to multiplication with the value vector and finally aggregated to yield the ultimate output vector.

2. Multi-Head Self-Attention Mechanism

To effectively manage interdependencies among diverse positions, the transformer model has ingeniously incorporated a sophisticated multi-head self-attention mechanism. Specifically, distinct linear transformations are applied to the input vectors, thereby yielding a multitude of query vectors, key vectors, and value vectors. Then, self-attention calculation is performed on each attention head separately, and the ultimate output vector materializes through the concatenation of the output vectors from each respective head.

Assuming there are h attention heads, each with a dimension of d_k, the equation for the multi-head self-attention mechanism is depicted below:

$$head_i = Attention(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V),$$

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h) \cdot W^o,$$

where W_i^Q, W_i^K, and W_i^V are the linear transformation matrices corresponding to the i-th attention head, and W^o is the linear transformation matrix for the concatenated output vector. The hyperparameters of attention heads h and vector dimension d_k are subject to adjustment in accordance with the demands of the given task.

3. Encoder and Decoder

As for the encoder, given an input series $X = (x_1, \ldots, x_n)$, the output of the encoder $Z = (z_1, \ldots, z_n)$ can be calculated via the following equations:

$$Z_0 = X,$$

$$Z_i = LayerNorm(Z_{i-1} + MultiHead(Z_{i-1}, Z_{i-1}, Z_{i-1})),$$

$$Z_i = LayerNorm(Z_i + FeedForward(Z_i)),$$

where $LayerNorm(\cdot)$ is a function employed to normalize the input vector and $FeedForward(\cdot)$ is a feed-forward network layer that extracts features.

As for the decoder, considering the input $Y = (y_1, \ldots, y_m)$ and the results produced by the encoder, the decoder output $O = (O_1, \ldots, O_m)$ can be determined as follows:

$$O_0 = Y,$$

$$O_i = LayerNorm(O_{i-1} + MultiHead(O_{i-1}, O_{i-1}, O_{i-1})),$$

$$O_i = LayerNorm(O_i + MultiHead(O_i, Z, Z)),$$

$$O_i = LayerNorm(O_i + FeedForward(O_i)),$$

where the first two equations are the same as those of the encoder, while the third equation represents the encoder–decoder attention layer, which aligns the output vectors of the encoder and the decoder.

Appendix C

The definition of MAE, RMSE, and MAPE Formulation in Section 4.3 are presented in this section.

Mean absolute error (MAE), rooted mean squared error (RMSE), and mean absolute percentage error (MAPE) are three measures that quantify the variance between the anticipated and observed outcomes. MAE is calculated as the mean of the absolute differences between the predicted and actual values, which is a reliable measure of accuracy when the dataset has outliers or extreme values. The smaller the MAE, the better the accuracy of the model. RMSE is a frequently employed metric for quantifying the disparity between projected and factual data points. It is a reliable measure of accuracy when the data set does not have outliers. RMSE is always larger than MAE, and the smaller values of RMSE denote the higher accuracy of the model. MAPE is calculated as the average of the absolute differences between the prediction and ground truth divided by the latter. MAPE is a useful measure when comparing the accuracy of different models or forecasting methods. The smaller the MAPE, the better the performance of the model. However, MAPE can be misleading when the real values are close to zero or when there are extreme values in the dataset. Consequently, we add a positive value approaching 0 to the denominator of the equation to prevent it from being zero. Their equations are as follows:

$$MAE = \frac{1}{k}\sum_{i=1}^{k}|y_i - \hat{y}_i|,$$

$$RMSE = \sqrt{\frac{1}{k}\sum_{i=1}^{k}|y_i - \hat{y}_i|^2},$$

$$MAPE = \frac{1}{k}\sum_{i=1}^{k}\frac{|\hat{y}_i - y_i|}{|y_i| + \varepsilon} \times 100\%,$$

where k represents the number of samples, y_i is the label of sample i, $\hat{y}_i$ represents the prediction of sample i, and ε is a very small positive value added to ensure that the denominator is not 0.

Appendix D

A map of scenarios in the experimentation on the METR-LA dataset.

The METR-LA dataset is a subset extracted from the comprehensive Los Angeles County road network traffic dataset. This original traffic dataset encompasses approximately 8900 traffic loop-detector sensors deployed on highways, collecting information from about 2036 buses and 35 trains that traverse across 145 distinct routes within Los Angeles County. The METR-LA dataset consists of data selected from 207 sensors, spanning a period of four months from 1 March 2012 to 30 June 2012. The distribution of sensors in the METR-LA dataset is visually depicted in Figure A1 as follows:

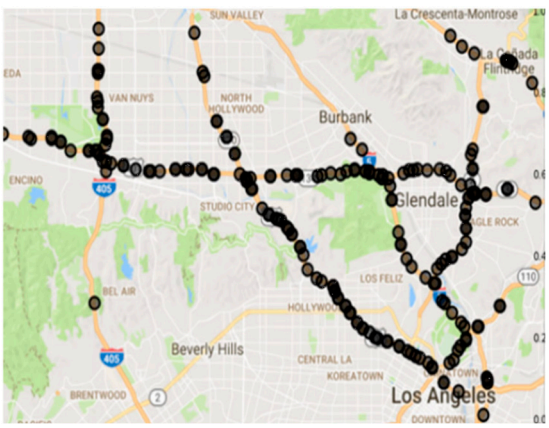

Figure A1. The distribution of the selected sensors in the METR-LA dataset.

References

1. Wang, F.Y. Parallel Control and Management for Intelligent Transportation Systems: Concepts, Architectures, and Applications. *IEEE Trans. Intell. Transp. Syst.* **2010**, *11*, 630–638. [CrossRef]
2. Li, Y.; Yu, R.; Shahabi, C.; Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In Proceedings of the Sixth International Conference on Learning Representations, Vancouver Convention Center, Vancouver, BC, Canada, 30 April–3 May 2018.
3. Wang, H.; Zhang, R.; Cheng, X.; Yang, L. Hierarchical Traffic Flow Prediction Based on Spatial-Temporal Graph Convolutional Network. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 16137–16147. [CrossRef]
4. Wang, Y.; Zheng, Y.; Xue, Y. Travel time estimation of a path using sparse trajectories. In Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 25–34.
5. Li, Y.; Zheng, Y.; Zhang, H.; Chen, L. Traffic Prediction in a Bike-Sharing System. In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Washington, DC, USA, 3–6 November 2015; pp. 1–10.
6. Cao, D.; Zeng, K.; Wang, J.; Sharma, P.K.; Ma, X.; Liu, Y.; Zhou, S. Bert-Based Deep Spatial-Temporal Network for Taxi Demand Prediction. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 9442–9454. [CrossRef]
7. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
8. Cho, K.; Gulcehre, B.V.M.C.; Bahdanau, D.; Schwenk, F.B.H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the EMNLP 2014: Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014.
9. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]
10. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and deep locally connected networks on graphs. In Proceedings of the Second International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
11. Passalis, N.; Tefas, A.; Kanniainen, J.; Gabbouj, M.; Iosifidis, A. Deep adaptive input normalization for time series forecasting. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 3760–3765. [CrossRef] [PubMed]
12. Arik, S.O.; Yoder, N.C.; Pfister, T. Self-adaptive forecasting for improved deep learning on non-stationary time-series. *arXiv* **2022**, arXiv:2202.02403. [CrossRef]
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Thirty-First Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; p. 30.
14. Wu, C.H.; Ho, J.M.; Lee, D.T. Travel-Time Prediction with Support Vector Regression. *IEEE Trans. Intell. Transp. Syst.* **2004**, *5*, 276–281. [CrossRef]
15. Hamed, M.M.; Al-Masaeid, H.R.; Said, Z.M. Short-Term Prediction of Traffic Volume in Urban Arterials. *J. Transp. Eng.* **1995**, *121*, 249–254. [CrossRef]
16. Zambrano-Martinez, J.L.; Calafate, C.T.; Soler, D.; Cano, J.-C.; Manzoni, P. Modeling and Characterization of Traffic Flows in Urban Environments. *Sensors* **2018**, *18*, 2020. [CrossRef] [PubMed]
17. Van Hinsbergen, C.P.; Schreiter, T.; Zuurbier, F.S.; Van Lint, J.W.C.; Van Zuylen, H.J. Localized extended kalman filter for scalable real-time traffic state estimation. *IEEE Trans. Intell. Transp. Syst.* **2011**, *13*, 385–394. [CrossRef]
18. Cheng, J.; Li, G.; Chen, X. Research on Travel Time Prediction Model of Freeway Based on Gradient Boosting Decision Tree. *IEEE Access* **2019**, *7*, 7466–7480. [CrossRef]
19. Kong, X.; Chen, Q.; Hou, M.; Rahim, A.; Ma, K.; Xia, F. RMGen: A tri-layer vehicular trajectory data generation model exploring urban region division and mobility pattern. *IEEE Trans. Veh. Technol.* **2022**, *71*, 9225–9238. [CrossRef]
20. Tang, J.; Liu, F.; Zou, Y.; Zhang, W.; Wang, Y. An improved fuzzy neural network for traffic speed prediction considering periodic characteristic. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 2340–2350. [CrossRef]
21. Huang, W.; Song, G.; Hong, H.; Xie, K. Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 2191–2201. [CrossRef]
22. Ma, X.; Tao, Z.; Wang, Y.; Yu, H.; Wang, Y. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transp. Res. Part C Emerg. Technol.* **2015**, *54*, 187–197. [CrossRef]
23. Liu, J.; Zheng, F.; Liu, X.; Guo, G. Dynamic traffic flow prediction based on long-short term memory framework with feature organization. *IEEE Intell. Transp. Syst. Mag.* **2021**, *14*, 221–236. [CrossRef]
24. Zhang, J.; Zheng, Y.; Qi, D. Deep spatio-temporal residual networks for citywide crowd flows prediction. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31, pp. 1655–1661.
25. Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; Li, Z. Deep multi-view spatial-temporal network for taxi demand prediction. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32, pp. 2588–2595.
26. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Zhang, C. Graph wavenet for deep spatial-temporal graph modeling. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 1907–1913.
27. Li, M.; Zhu, Z. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; Volume 35, pp. 4189–4196.

28. Cai, L.; Janowicz, K.; Mai, G.; Yan, B.; Zhu, R. Traffic transformer: Capturing the continuity and periodicity of time series 636 for traffic forecasting. *Trans. GIS* **2020**, *24*, 736–755. [CrossRef]
29. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In Proceedings of the Thirty Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; p. 29.
30. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
31. Atwood, J.; Towsley, D. Diffusion-convolutional neural networks. In Proceedings of the Thirty Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; p. 29.
32. Gilmer, J.; Schoenholz, S.; Riley, P.; Vinyals, O.; Dahl, G. Neural message passing for quantum chemistry. In Proceedings of the Thirty-Fourth International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 1263–1272.
33. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the Twenty-Eighth Conference on Neural Information Processing Systems, Montréal, QC, Canada, 8–13 December 2014; p. 27.
34. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; Zhang, C. Connecting the dots: Multivariate time series forecasting with graph neural networks. In Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Online, 23–23 August 2020; pp. 753–763.
35. Zhang, Q.; Chang, J.; Meng, G.; Xiang, S.; Pan, C. Spatio-temporal graph structure learning for traffic forecasting. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 1177–1185.
36. Wu, X.; Zhang, D.; Guo, C.; He, C.; Yang, B.; Jensen, C.S. AutoCTS: Automated correlated time series forecasting. In Proceedings of the 47th International Conference on Very Large Data Bases, Copenhagen, Denmark, 16–20 August 2021; Volume 15, pp. 971–983.
37. Huang, X.; Wang, J.; Lan, Y.; Jiang, C.; Yuan, X. MD-GCN: A multi-scale temporal dual graph convolution network for traffic flow prediction. *Sensors* **2023**, *23*, 841. [CrossRef] [PubMed]