Trafformer: Unify Time and Space in Traffic Prediction

Di Jin[1,2], Jiayi Shi[1], Rui Wang[1], Yawen Li[3,*], Yuxiao Huang[4], Yu-Bin Yang[2]

[1]College of Intelligence and Computing, Tianjin University, Tianjin, P.R. China
[2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, P.R. China
[3]School of Economics and Management, Beijing University of Posts and Telecommunications, Beijing, P.R. China
[4]Columbian College of Arts Sciences, George Washington University, Washington, D.C., USA
{jindi,shijiayi,wr1895}@tju.edu.cn, warmly0716@bupt.edu.cn, yuxiaohuang@gwu.edu, yangyubin@nju.edu.cn

Abstract

Traffic prediction is an important component of the intelligent transportation system. Existing deep learning methods encode temporal information and spatial information separately or iteratively. However, the spatial and temporal information is highly correlated in a traffic network, so existing methods may not learn the complex spatial-temporal dependencies hidden in the traffic network due to the decomposed model design. To overcome this limitation, we propose a new model named Trafformer, which unifies spatial and temporal information in one transformer-style model. Trafformer enables every node at every timestamp interact with every other node in every other timestamp in just one step in the spatial-temporal correlation matrix. This design enables Trafformer to catch complex spatial-temporal dependencies. Following the same design principle, we use the generative style decoder to predict multiple timestamps in only one forward operation instead of the iterative style decoder in Transformer. Furthermore, to reduce the complexity brought about by the huge spatial-temporal self-attention matrix, we also propose two variants of Trafformer to further improve the training and inference speed without losing much effectivity. Extensive experiments on two traffic datasets demonstrate that Trafformer outperforms existing methods and provides a promising future direction for the spatial-temporal traffic prediction problem.

Introduction

With the trend of global urbanization and population growth, transportation systems are growing bigger and more complex, making Intelligent Transportation System (ITS) (Jabbarpour et al. 2018) become increasingly important. Traffic prediction is one of the most challenging tasks in ITS. By learning historical traffic conditions and patterns, accurate traffic prediction helps with a wide range of traffic management problems, such as traffic congestion, travel duration, and controlling flow, etc.

Some traffic forecasting works treat traffic prediction as a general time series forecasting problem, including classic time series models, e.g. auto-regressive and integrated moving average (ARIMA) models; machine learning and deep learning models, e.g. recurrent neural networks (RNNs) (Huang et al. 2014; Fu, Zhang, and Li 2016), one-dimensional convolutional neural networks (1D-CNNs) (Stoller et al. 2019), and Transformer varients (Kitaev, Kaiser, and Levskaya 2020; Zhou et al. 2021; Wu et al. 2021). Although these models account for temporal dependencies in traffic data and have gained success in some traffic prediction tasks, they neglect spatial correlations, compromising the model performance in accuracy and other aspects. The traffic flow on a road is not only influenced by its historical traffic conditions but also the conditions of upstream and downstream roads, making traffic prediction more challenging than other time series forecasting problems.

To better utilize the space information hidden in traffic data, some efforts have been dedicated to combine time series models with 2D-CNNs or GNNs. Some models, like (Zhang, Zheng, and Qi 2017) and (Cao et al. 2017), convert traffic data into spatial-temporal grids and then treat these grid-style spatial dependencies as images and impose two-dimentional convolutional neural network (2D-CNN) on the converted grid traffic data. A more natural way to utilize space information in traffic data is to treat it as a series of spatial-temporal graphs. These works use RNN, 1D-CNN or Transformer for encoding temporal information and GNNs, like GCN, GAT (Yu, Yin, and Zhu 2019; Guo et al. 2020), for encoding spatial information. For example, DCRNN (Li et al. 2017) constructs traffic graph by nodes proximity (measures road network distance of sensor nodes) and then replaces 2D-CNNs with GCNs in order to extract graph-style spatial dependencies, in which spatial dependencies are modeled as a diffusion process. PVCGN (Liu et al. 2020) constructs a physical graph, a similarity graph and a correlation graph based on the physical topology of a metro system and human domain knowledge, and then incorporates these extracted graphs into a Graph Convolutional Gated Recurrent Unit (GC-GRU) for local spatial-temporal representation learning.

However, existing methods have a common limitation preventing them from getting better performance, that is, all of them encode time and space information separately. To be specific, as shown in Fig.1, for an input traffic flow graph with multiple timestamps, these methods either encode time-series first, encode space first, or iteratively deal with time and space. These strategies can use both time and

*Corresponding author

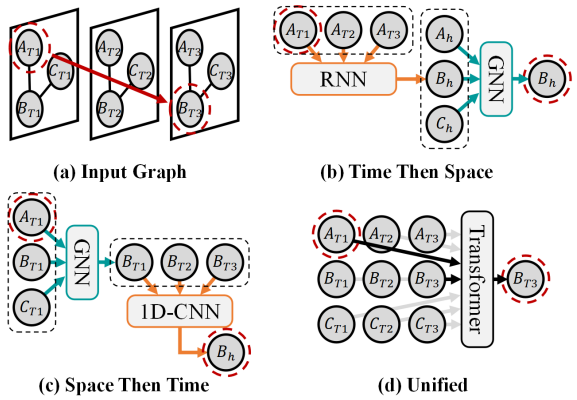

Figure 1: (a) Input traffic flow graph with multiple time stamps. (b), (c) and (d) are three different ways to encode temporal and spatial information.

space information to some extent. If we look at a node pair in the same timestamp or the same physical node, information seems to flow well. However, if we try to analyze a node pair in different timestamps and positions in the spatial-temporal graph series, we can observe the following two phenomena: First, the distance between this node pair is at least 2-hops. Second, when feeding into the encoder, information is already highly mixed before it reaches the target node, especially when this pair of nodes are more than 2-hops away. Such issues prevent the models discussed earlier from learning spatial-temporal correlation hidden in the traffic data, which is exactly highly spatial-temporal correlated.

To solve this problem, we propose the following design principles that an effective model for traffic prediction should follow: First, instead of encoding time and space information separately, this model should unify time and space, encoding them in one module. Second, this model should let message pass freely in the learning process to let correlated nodes from different timestamps and positions interact with each other and truly catch the complex spatial-temporal dependencies in the traffic data. Following these design principles, we propose a model named Trafformer. First, Trafformer unifies time and space by introducing a spatial-temporal correlation matrix into the framework which combines all timestamps of all nodes and then feeds them into a transformer-style encoder-decoder architecture with three kinds of position encoding: time encoding, space encoding, and feature encoding. Then, to allow arbitrarily correlated nodes to interact freely, the attention mechanism of Trafformer works in a fully connected manner in the encoder and decoder respectively. Following the same design principle, we use the generative style decoder to predict multiple timestamps in only one forward operation instead of the iterative style decoder in classic Transformer. Finally, to let Trafformer works more efficiently, we also give two sparsed version of Trafformer, which can reduce the complexity brought about by the full spatial-temporal self-attention graph as a trade-off between efficiency and precision.

Preliminary

A traffic network can be represented as a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ is the set of N nodes representing the sensors, $\mathcal{E}$ is the set of edges reflecting physical connectivity between sensors, and $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix storing the nodes proximity or distance in the network. $X^t \in \mathbb{R}^{N \times f}$ denotes the feature matrix of the graph that is observed at time t, where f is the number of features.

Traffic forecasting aims to predict future traffic from a series of historical traffic observations detected by sensors on a road network. In this paper, we focus on forecasting traffic speeds $v^t \in \mathbb{R}^N$ at time step t for the N sensors, and volume and density can be similarly calculated as traffic speeds. We formalize forecasting as learning a traffic forecasting model $\mathcal{F}$ that predicts T future traffic conditions through the given M historical traffic conditions on the premise of a network $\mathcal{G}$:

$$\hat{v}_{t+1}^{t+T} = \mathcal{F}(v_{t-M+1}^{t}; \mathcal{G}) \tag{1}$$

where v_i^{i+n} denotes an array of feature matrices from time stamp i to $i+n$: $[v^i, v^{i+1}, \cdots, v^{i+n}]$.

The Model

Overview

To let the model truly unify information from time and space, here we propose a novel approach which mainly includes three parts. 1) Position Encoding, which integrates spatio-temporal information into a model from three aspects: space encoding, time encoding, and feature encoding. 2) Encoder, which realizes the free interaction of different nodes at different moments through the stacked self-attention layer. 3) Decoder, which adopts the generative inference to predict all time slices of all nodes at once. The architecture of our proposed Trafformer model is shown in Fig.2. To further reduce the computation complexity brought about by the spatial-temporal graph, two sparsification strategies are given to improve the model efficiency. In the following sections, we will introduce the details of each component.

Position Encoding for Spatial Temporal Graph

Traffic data is highly spatio-temporal related. In order to better process the temporal and spatial information of traffic data, we design a position encoding method to inject the temporal and spatial information into the model.

Time Encoding. This module can encode time information in two ways. We can think of a sequence as points on the sine or cosine curve from front to back, so we use the sine and cosine functions of different frequencies:

$$\begin{aligned} E_{(time,2i)} &= \sin(time/10000^{2i/D}) \\ E_{(time,2i+1)} &= \cos(time/10000^{2i/D}) \end{aligned} \tag{2}$$

where T is the number of timestamps, and D is the dimension of hidden layers. The time information after encoding can be expressed as $E_{time} \in \mathbb{R}^{1 \times T \times D}$. The dimension representing the number of nodes is 1, because spatial information is the same across different timestamps, that is, every node in one timestamp is the same. In addition, time encoding can also be learned from data.

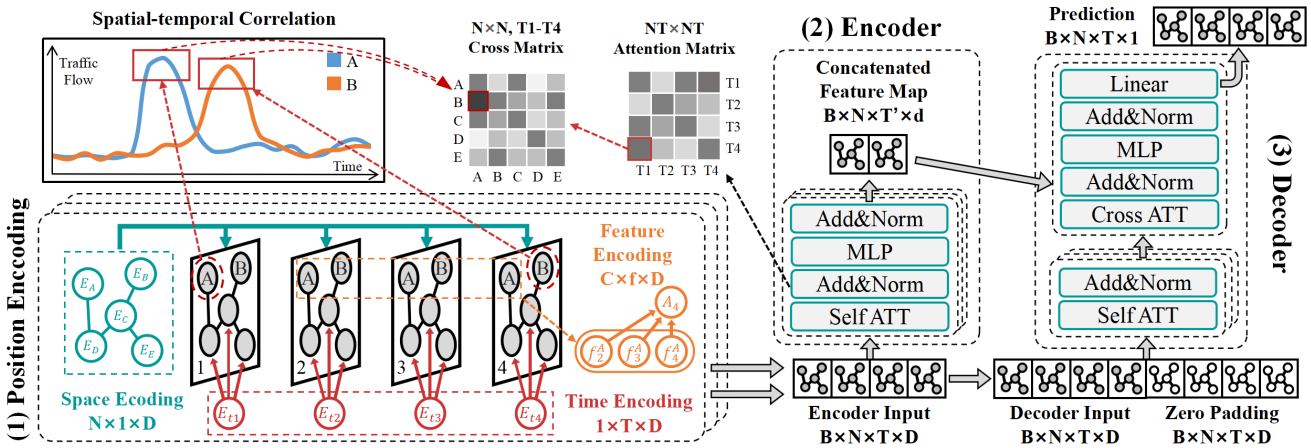

Figure 2: The structure of Trafformer, consisting of three components: (1) position encoding integrates spatio-temporal information into the model, (2) encoder realizes the free interaction of different nodes at different moments through the stacked self-attention layer, and (3) decoder adopts transductive learning method to predict all time slices of all nodes at once.

Space Encoding. A position embedding vector is used in the Transformer layer to encode the relative distance between any two positions. The nodes in graph exist in a multi-dimensional space, connected by edges, and are not arranged sequentially. Therefore, for each node in the traffic network, there is a learnable embedding to describe the spatial location information between nodes. This module encodes spatial location information.

We use the structure information of the graph to pre-calculate the Laplacian eigenvectors (Belkin and Niyogi 2003) and take them as the position information of the nodes. We use the k smallest non-trivial eigenvectors of node as E_{space}. The process of calculating the eigenvectors by factoring the Graph Laplacian matrix can be represented as:

$$\Delta = \mathrm{I} - D^{-1/2} A D^{-1/2} = U^T \Lambda U, \tag{3}$$

where D is the degree matrix, Λ and U correspond to the eigenvalues and eigenvectors respectively. After encoding the spatial information can be expressed as $E_{space} \in \mathbb{R}^{N\times 1\times D}$. Each node has an embedding whose time dimension is 1, indicating that each time slice of a node in the graph are the same.

Feature Encoding. Data on traffic flow can change significantly over time due to events such as roadworks and extreme weather, so the data observed by a sensor is heavily influenced by its surroundings. Due to the spatio-temporal correlation, each node can affect the state of other nodes at different times. However, in the traditional self-attention layers of Transformer, local context such as graph structure is not fully utilized and the interaction between various points is ignored. Therefore, we use feature encoding to improve the efficiency of self-attention and then improve the prediction ability of the model.

The causal convolution (Li et al. 2019) of kernel size k and stride 1 is used to process the input data in Feature Encoding, and their similarity is calculated by their local context information. The output of feature encoding can be expressed as $E_{feature} \in \mathbb{R}^{C\times f\times D}$, where C is the number of convolution kernel. As shown in orange in Fig.2, the attributes of node A at time 1, time 2, and time 3 are aggregated back to node A. This kind of location perception can match the most relevant features according to the shape of the graph. As shown in the red box, it can observe the pattern of traffic flow peaks in different areas, which is helpful for accurate prediction.

Encoder for Spatial Temporal Graph

In the previous traffic prediction models using Transformer (Vaswani et al. 2017), the time information and the space information are encoded separately. For example, the input for the Transformer encoder of each node is $X \in \mathbb{R}^{L\times D}$ for encoding the temporal information. However, this separated design is not enough for the learning of the complex spatial-temporal correlation hidden in the traffic network. Trafformer, on the other hand, integrates the time and space information into a transformer-style model. The input of the encoder-decoder structure is obtained by adding three positional codes, namely time coding, space coding, and feature coding:

$$X_{input} = E_{time} + E_{space} + E_{feature} \tag{4}$$

where $X_{input} \in \mathbb{R}^{B\times N\times T\times D}$, and B is batchsize.

Encoder is composed of l identical layers. In each layer, the attention mechanism works in a fully connected way. We take X_{input} as the input of self-attention. Through linear transformation, we obtain three matrices Q, K and V. The multi-head attention layer we use is similar to that in Transformer, Q_i, K_i, and V_i for i-th head can be calculated as:

$$[Q_i, K_i, V_i] = X[W_i^Q, W_i^K, W_i^V] \tag{5}$$

where W_i^Q, W_i^K, $W_i^V \in \mathbb{R}^{D\times D}$. The calculated output matrix of the attention layer of the i-th head is:

$$\mathrm{Attention}(Q_i, K_i, V_i) = \mathrm{softmax}(\frac{Q_i K_i^\top}{\sqrt{D_i}})V_i \tag{6}$$

The multi-head attention mechanism allows each node on each timestamp to interact with each node on other timestamps, so as to realize the free interaction of any relevant nodes and capture the complex spatio-temporal correlation in the traffic network. Thus, for an h-th head attention layer, the operation can be expressed as:

$$X_{multi} = \text{Concat}(\text{Attention}(Q_1, K_1, V_1), \\ ..., \text{Attention}(Q_h, K_h, V_h))W^O \quad (7)$$

We construct a spatio-temporal connection graph $A_{ST} \in \mathbb{R}^{NT \times NT}$ formed by $T \times T$ adjacency matrix A, as shown in the Attention Matrix in Fig.2, representing the interaction relationship between each node and all other nodes at each timestamp. A_{ST} is used for graph convolution operations. We use residual connection (He et al. 2016) in each layer separately. Adding the input results and output results of each layer, which is used to solve the problem of multi-layer network training, so that the network can only focus on the part of the current difference. Then we use layer normalization (Ba, Kiros, and Hinton 2016) to speed up convergence, that is, the output of each sublayer is $\text{LayerNorm}(X + \text{sublayer}(X))$, where $\text{sublayer}(X)$ is a function implemented by the sublayer itself.

The feature map of encoder is obtained by the fully connected self-attention mechanism, and then the distilling operation (Zhou et al. 2021) is used to give higher weight to the dominant feature with dominant attention, that is, the pooling operation of one-dimensional convolution is performed first to obtain the concatenated feature map. The process of distilling operation from i to $i+1$ layer is as follows:

$$X_{i+1}^t = \text{Pooling}\left(\text{ELU}(\text{Conv1d}([X_i^t])) \right) \quad (8)$$

where $[\cdot]$ represents the key operation in the fully connected self-attention layer, Conv1d represents the one-dimensional convolution operation. The pooling operation is used to distill the useful information and reduce the whole memory usage at the same time. It maps the time dimension from T to T'. The output of the last encoder is the concatenated feature map, which is denoted as $X_{featuremap} \in \mathbb{R}^{B \times N \times T' \times D}$.

Decoder for Spatial-temporal Graph

Traditional Transformer has two stacked multi-head attention layers in decoder, which is a dynamic decoding process, that is, iterative output step by step. Informer makes an improvement in this aspect. It uses batch generative prediction to directly output multi-step prediction results, and predicts all time slices of a node at a time. Trafformer uses Transductive learning like Informer, taking the entire graph as input. Moreover, it goes above and beyond such that multiple timestamps are predicted in one forward operation, and all time slices of all nodes are predicted at one time, thus the accuracy of predicting spatial-temporal output is further improved. The Decoder's input is:

$$X_{decoder} = \text{Concat}(X_{token}, X_0) \quad (9)$$

where $X_{decoder} \in \mathbb{R}^{B \times N \times (T+T) \times D}$. X_0 represents the placeholder for the forecast sequence and X_{token} is the start token, with $X_0, X_{token} \in \mathbb{R}^{B \times N \times T \times D}$.

Strategies to Reduce Complexity

In Trafformer, the complexity of self-attention reaches $O(N^2T^2)$. In this part, we will try to optimize the computational efficiency of self-attention and reduce its complexity.

The Informer Strategy. Some researchers have visualized the dot product results in self-attention. Finding that long tail distribution (Yang and Xu 2020; Zhang et al. 2021) in characteristic graph and verified the sparsity of the self-attention mechanism: an element in a sequence generally has high similarity with only a few elements. That is, the dot product of a few querys and keys computations dominate the probability distribution.

After each query is dotted with all keys in the sequence, the discrete distribution obtained by softmax is different. If the distribution of a query is similar to the uniform distribution, then self-attention becomes the trivial sum of values, which has no reference value. However, if the probability value obtained by a query is very different from the uniform distribution, there must be a few larger probability values, which are obviously more meaningful. Therefore, the core idea of algorithm optimization is to find these important queries.

The authors of Informer simplified the calculation of the measurement results of query sparsity (Tsai et al. 2019). We use this calculation, but unlike Informer input sequence, the input of Trafformer are graphs, so $Q, K, V \in \mathbb{R}^{D \times D}$, and q_i, k_i, v_i stand for their i-th row respectively. The simplified calculation can be expressed as:

$$\overline{M}(q_i, K) = \max_j \{ \frac{q_i k_j^\top}{\sqrt{d}} \} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{q_i k_j^\top}{\sqrt{d}} \quad (10)$$

where the first term is the simplified calculation method of Log-Sum-Exp (LSE) of q_i on all the keys, and the second term is their arithmetic mean.

If the i-th query obtains a larger $\overline{M}(q_i, K)$, its attention probability is more discriminative and is more likely to include the major dot product pairs in the head region of the long tail self-attention distribution.

However, it is infeasible to calculate sparsity for each query due to the computational cost. Therefore, in practice, the key part of each query is randomly sampled first, and then the sparsity score is calculated. The n queries with the highest score are selected to calculate the dot product between them and all keys, and then the attention result is obtained. The input of the self-attention layer is directly averaged as the output. The overall time complexity of this simplified computation is $O(NTln(NT))$.

The Sparse Graph Strategy. As mentioned in the section Encoder for Spatial Temp, since the information of the traffic network is transmitted simultaneously in time and space dimensions, we constructed a spatio-temporal connection graph $A_{ST} \in \mathbb{R}^{NT \times NT}$ formed by $T \times T$ adjacency matrix A. Therefore, it is equivalent to that all time slices of each node and all neighbor nodes are connected, the spatio-temporal correlation between nodes at different times under different timestamps can be fully considered.

However, this also introduces some problems. First, the dimension of A_{ST} is very high, which will bring consid-

erable computational load to the model. Here, we try to find a way to reduce the computational load and improve the operation efficiency of the model. Second, each node has a different impact on its neighbors. The adjacency matrix contains only 0 and 1, so the weight calculation of the self-attention mechanism may be limited. If two nodes in a spatio-temporal connection graph are connected, even if they are not actually related for a period of time, or their degree of correlation is very low, their features will be considered to be related and given a weighted sum. Conversely, if two nodes in a spatio-temporal connection graph are not connected, they will not be given a weighted sum even if they are strongly correlated in reality.

Therefore, referring to the simplification method of STSGCN (Song et al. 2020), we replace A_{ST} from the fully connected matrix with a sparse matrix A_{spare} to adjust the weight of the calculation.In this way, the correlation between different nodes can be more reasonably emphasized in the calculation of weights, and also reduce the computation load and improve the operation efficiency of the model. The sparse matrix is calculated as:

$$A_{mask} = \mathbb{I} \otimes A^k \tag{11}$$

where $\mathbb{I} \in \mathbb{R}^{T \times T}$ is an all-ones matrix, $\otimes$ stands for Kronecker product. k is the neighbor order in the spatial direction, that is, the matrix to the K-th power. In our experiment we tried the values of k of 1 and 2. $A_{mask} \in \mathbb{R}^{N \times N}$ indicates the mask matrix. After the self-attention matrix is masked, the sparse matrix A_spare is obtained. The formula is as follows:

$$A_{sparse} = A_{mask} \odot (Q * K) \tag{12}$$

where $A_{sparse} \in \mathbb{R}^{NT \times NT}$, and $\odot$ stands for Hadamard product. This sparse matrix A_{sparse} can be used instead of the spatio-temporal connection graph A_{ST} to reduce the operation overhead. Using the spatio-temporal graph locality assumption, the complexity is reduced to $O(E \times T^2)$ in this section.

We choose Mean Squared Error (MSE) as the loss function, which can evaluate the degree of data change. The smaller the MSE, the higher the accuracy of the prediction model in describing the experimental data.

$$L(\hat{\mathbf{X}}^{(t+1):(t+T)}; \mathbf{\Theta}) = \frac{1}{TN} \sum_{i=1}^{i=T} \sum_{j=1}^{j=N} (\hat{\mathbf{X}}_j^{(t+i)} - \mathbf{X}_j^{(t+i)})^2 \tag{13}$$

where $\mathbf{\Theta}$ is the model parameter.

Experiments

In this section, we compare the performance of Trafformer with state of the art models in terms of traffic flow prediction using two real-world datasets. We designed ablation experiments and evaluated the computational efficiency of different complexity reduction strategies.

Experimental Setup

Datasets. We evaluate the performance of our proposed model using two real-world datasets, the public transportation network datasets METR-LA and PEMS-Bay. METR-LA recorded traffic speed statistics by loop detectors from 207 sensors on Los Angeles County freeways for four months, from March 1, 2012, to June 30, 2012. PEMS-BAY is collected from loop detectors on Los Angeles County freeways and contains six months of traffic speed information from 325 sensors in the BAY area from January 1, 2017, to May 31, 2017. During the experiment, both datasets are sorted in ascending chronological order and are split into three parts for training, validation, and testing, which account for 70%, 10%, and 20% respectively. For a fair comparison, all methods share the same random split. Table 1 provides detailed data set statistics.

Data	Nodes	Edges	Time Steps	MissingRatio
METR-LA	207	1515	34272	8.109%
PEMS-BAY	325	2369	52116	0.003%

Table 1: Summary statistics of METR-LA and PEMS-BAY.

Baselines. We compare Trafformer with the following models:

- ARIMA: Auto-Regressive Integrated Moving Average (Li et al. 2017) is a classical time series mathematical model.
- AGCRN: Adaptive Graph Convolutional Recurrent Network (Bai et al. 2020) combines GRU.
- WaveNet: Inflated causal convolution is used to acquire receptive fields (Oord et al. 2016).
- Graph Wavenet: Graph WaveNet (Wu et al. 2019) conducts graph convolution with adaptive adjacency matrix.
- Transformer: solves the long-distance dependence problem based on self-attention structure (Vaswani et al. 2017).

Parameter Setup. For ARIMA, AGCRN, WaveNet and GraphWaveNet, we set their parameters as what were used by their authors. For Transformer and Trafformer, we set the encoder layer to 2 and decoder layer to 1. The hidden dimension for each attention layer is set to 32. For Trafformer, the batch size for METR-LA is set to 11 and the batch size for PEMS-Bay is set to 5. All learnable parameters are initialized with a normal distribution. We use Adam optimizer with an initial learing rate of 0.002. We test 3 settings for dropout rate (0, 0.05, 0.2) and set the rate to 0. We choose evaluation metrics including mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE).

Results of Traffic Flow Prediction

Table 2 compares the performance of Trafformer and different methods of traffic flow prediction task on the two datasets for 15 min, 30 min, and 60 min advance prediction.

In terms of short-range prediction, that is, 15-minute prediction, the prediction performance of Trafformer is comparable with that of Graph Wavenet. Both models are the optimal or suboptimal results, indicating that our model Trafformer has reached an advanced level in short-range prediction. Trafformer delivers the best results for long-range

Data	Models	15 min			30 min			60 min		
		MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
METR-LA	ARIMA	3.99	8.21	9.60	5.15	10.45	12.70	6.90	13.23	17.40
	AGCRN	3.67	9.58	8.45	4.75	12.1	10.77	6.13	14.86	13.46
	WaveNet	2.99	5.89	8.04	3.59	7.28	10.25	4.45	8.93	13.62
	Graph Wavenet	**2.70**	**5.19**	**6.97**	3.12	6.32	**8.47**	3.55	7.39	10.02
	Transformer	3.08	6.05	8.35	3.74	7.56	10.79	4.78	9.52	14.68
	Trafformer	2.78	5.35	7.32	**3.05**	**6.18**	8.67	**3.41**	**7.17**	**9.96**
PEMS-BAY	ARIMA	1.62	3.30	3.50	2.33	4.76	5.40	3.38	6.50	8.30
	AGCRN	1.39	2.98	X	1.76	3.97	X	2.14	4.85	X
	WaveNet	1.39	3.01	2.91	3.59	7.28	10.25	4.45	8.93	13.62
	Graph Wavenet	1.33	**2.82**	**2.77**	1.64	**3.70**	**3.70**	2.01	4.56	4.67
	Transformer	2.21	4.43	5.15	2.28	4.59	5.32	2.39	4.86	5.65
	Trafformer	**1.31**	2.83	2.92	**1.61**	3.74	3.82	**1.88**	**4.38**	**4.59**

Table 2: Performance comparison of Trafformer and other baseline models on traffic flow prediction task. The lower the value the better the model. The best result is in bold and the second best is underlined. X means the result is too large to be included in the table.

Method	MAE	RMSE	ATT	AIT	Complexity
Trafformer	3.41	7.17	952.35	67.17	$O(NT^2)$
Trafformer-IS	4.06	8.07	372.20	14.38	$O(NTln(NT))$
Trafformer-A	4.13	7.73	670.78	28.62	$O(ET^2)$
Trafformer-A2	3.64	7.60	785.21	30.17	$O(ET^2)$

Table 3: Different complexity reduction strategies. ATT: Average training time, AIT: Average inference time

prediction, especially for 60 minutes. For the dataset MetR-LA, Trafformer has better MAPE values than ARIMA, AGCRN, Wavenet, Graph Wavenet, and Transformer models by 7.44%, 3.5%, 3.66%, 0.06%, and 4.72% respectively, which is significantly better than all of the other models. For the experimental results on data set EMS-Bay, the MAPE values of Trafformer are also 3.71%, 9.03%, 0.08%, and 1.06% better than those of the rival models. These experimental results show that our model greatly improves the accuracy of long-term traffic flow prediction.

Computation Efficiency

We compare the experimental training time and theoretical complexity of different complexity-reducing strategies. The results are given in Table 3. We can see that both strategies for reducing complexity improve training efficiency and inference efficiency. For efficiency comparison, the Informer strategy achieves the best efficiency among all variants, but it also gets the worst performance. Higher informer sample factor can trade efficiency for performance. The sparse graph strategy does not perform well when we only concatenate original adjacency matrix A to build spatial-temporal graph. But when we concatenate A^2 to build the spatial-temporal graph, it performs much better thanks to a bigger receptive field in space dimension. Moreover, the training time is consistent with the theoretical analysis we give in Section Strategies to Reduce Complexity.

Ablation Study

To verify the effectiveness of designs in Trafformer, we conduct ablation experiments on METR-LA. Table 4 shows metric of MAE, RMSE, and MAPE. Here is the conclusion.

- Without space encoding, Trafformer can't get the spatial information of each node, so it fails to learn how to use the information of neighbor nodes because it can't distinguish them in the same timestamp. We can see that Trafformer without space encoding gets performance similar to Transformer.
- Without the spatial-temporal graph, Trafformer works like a Transformer with space encoding and feature encoding but no information from neighbor nodes. However, this configuration still works better than Transformer, which proves the effectiveness of the design of feature encoding.
- Trafformer with iterative decoder performs well in short-term prediction, but it gets much worse when the prediction window is longer. On the contrary, Trafformer with generative decoder performs stable with the offset increasing. It proves the generative decoder works better for learning spatial-temporal dependency between arbitrary outputs and avoiding error accumulation.

Related Work

Spatial-Temporal Graph

In recent years, graph neural networks (GNNs) have become the frontier of deep learning research, showing state-of-the-art performance in various applications (Wu et al. 2020b; Jin et al. 2021a; Yu et al. 2021). With strong learning ability, CNN can capture the hidden spatial information in the non-Euclidean structure data in the traffic field and identify the network structure (Jin et al. 2021b), which brings new opportunities to solve the problem of traffic prediction. The majority of Spatial-temporal Graph Networks follow two directions, RNN-based and CNN-based approaches.

Models	15 min			30 min			60 min		
	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
No E_{space}	3.14	6.15	8.55	3.8	7.63	10.86	4.88	9.65	14.68
No A_{ST}	3.01	5.93	8.11	3.30	7.14	9.22	4.25	8.73	13.19
Trafformer Iterative	<u>2.80</u>	<u>5.48</u>	**7.31**	<u>3.27</u>	<u>6.99</u>	<u>9.04</u>	<u>3.91</u>	<u>7.57</u>	<u>11.75</u>
Trafformer	**2.78**	**5.35**	<u>7.32</u>	**3.05**	**6.18**	**8.67**	**3.41**	**7.17**	**9.96**

Table 4: Ablation experiments on METR-LA. The best result is bold and the second best is underlined.

The rise of GNNs inspired the following work to employ spatial-temporal GNNs (STGNNs) in traffic prediction (Guo et al. 2019; He, Chow, and Zhang 2018), in which both spatial and temporal information are utilized. STGCN (Yu, Yin, and Zhu 2017) stacks multiple spatial-temporal convolution blocks and each block concatenates two temporal convolutions and one graph convolution layer. In STSGCN (Song et al. 2020), a localized spatial-temporal graph that includes both temporal and spatial attributes is constructed first and a spatial-based GCN method is then applied. RiskOracle (Zhou et al. 2020a) increases the prediction granularity to the level of minutes. RiskSeq (Zhou et al. 2020b) predicts sparse urban accidents with finer granularity and multiple steps from a spatio-temporal perspective. By defining and using out-degree and in-degree matrices, Graph WaveNet (Xu and Liu 2021) assembles graph convolution with dilated casual convolution, each graph convolution layer tackles spatial dependencies of nodes information extracted by dilated casual convolution layers at different granular levels. DCRNN (Li et al. 2017) models the bidirectional diffusion process to capture the influence of both upstream and downstream traffic. GGRU (Zhang et al. 2018) expands on DCRNN, proposing a unified method for constructing a recurrent neural network based on an arbitrary graph aggregator. AGCRN (Bai et al. 2020) captures fine-grained spatial and temporal correlations in traffic series automatically based on the recurrent networks. Temporal correlation and spatial correlation are closely intertwined in real life. The future trend is to combine CNNs/GCNs with RNNs to extract spatial and temporal correlation of traffic prediction.

Time Series Forecasting

Machine learning models for traffic prediction driven by big data have attracted great attention in academia and industry in recent years (Dai et al. 2019; Yao et al. 2019; Zhao et al. 2020). Due to the immense importance of time series forecasting, various models have been well developed. Most deep learning models for traffic forecasting have been built by utilizing RNNs due to their ability to memorize temporal dependencies in time series via self-circulation. RNN and GRUNN (Fu, Zhang, and Li 2016) compared different RNN models, namely LSTM and GRU, finding that when capturing forward temporal dependencies, GRU achieved better performance than LSTM in forecasting traffic. SUB-LSTM (Cui, Ke, and Wang 2018) proposed a deeply stacked bidirectional and unidirectional LSTM architecture that can capture both forward and backward dependencies in time series. DeepAR (Salinas et al. 2020) combines autoregressive methods and RNNs to model the probabilistic distribution of future series. LSTNet (Lai et al. 2018) uses CNN and RNN to extract short-term local dependency patterns among variables and to discover long-term patterns for time series trends. Attention-based RNNs (Shih, Sun, and Lee 2019) can model long-term dependency in time series data. These deep forecasting models mainly focus on the temporal relation modeling by recurrent connections, temporal attention, or causal convolution.

Recently, Transformer (Wu et al. 2020a; Vaswani et al. 2017) has been developed as a new architecture in deep learning, such as natural language processing (Devlin et al. 2018; Brown et al. 2020), audio processing (Huang et al. 2018) and even computer vision (Dosovitskiy et al. 2020; Liu et al. 2021), which employs attention mechanisms along with a position encoding strategy for sequence modeling. LogTrans (Li et al. 2019) introduces the local convolution to Transformer and proposes the LogSparse attention to select time steps following the exponentially increasing intervals. Reformer (Kitaev, Kaiser, and Levskaya 2020) replaces dot-product attention with one that uses locality-sensitive hashing and uses reversible residual layers instead of the standard residuals. Informer (Zhou et al. 2021) extends Transformer with KL-divergence based ProbSparse2 attention and achieves O(LlogL) complexity. Autoformer (Wu et al. 2021) is a decomposition architecture with inner decomposition blocks, which can empower the deep forecasting model with immanent progressive decomposition capacity. STTNs (Xu et al. 2020) focus on jointly modeling spatial and temporal dependencies by making full use of Transformer to solve traffic prediction tasks. Traffic Transformer (Cai et al. 2020) designs different temporal encoding strategies by explicitly modeling temporal patterns reflected in the time series.

Conclusion

In this paper, we propose the Traffromer model to learn the complex spatio-temporal correlation hidden in the traffic network. Specifically, Trafformer can unify spatio-temporal information in a model and predict multiple timestamps in a single forward operation with a generative decoder, to enable each node on each timestamp to interact with each node on other timestamps. In addition, we propose two variants of Trafformer to improve the training accuracy and speed. Experiments on two real datasets demonstrate the effectiveness of Trafformer algorithm in improving the ability of traffic prediction problems.

Acknowledgements

The work was supported by the National Natural Science Foundation of China (No. 62272340, 62276187, 62172056, 62176119)

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bai, L.; Yao, L.; Li, C.; Wang, X.; and Wang, C. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems*, 33: 17804–17815.

Belkin, M.; and Niyogi, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6): 1373–1396.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Cai, L.; Janowicz, K.; Mai, G.; Yan, B.; and Zhu, R. 2020. Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, 24(3): 736–755.

Cao, X.; Zhong, Y.; Zhou, Y.; Wang, J.; Zhu, C.; and Zhang, W. 2017. Interactive temporal recurrent convolution network for traffic prediction in data centers. *IEEE Access*, 6: 5276–5289.

Cui, Z.; Ke, R.; and Wang, Y. 2018. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. *CoRR*, abs/1801.02143.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J. G.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *ACL (1)*, 2978–2988. Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Fu, R.; Zhang, Z.; and Li, L. 2016. Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 324–328. IEEE.

Guo, K.; Hu, Y.; Qian, Z.; Liu, H.; Zhang, K.; Sun, Y.; Gao, J.; and Yin, B. 2020. Optimized graph convolution recurrent neural network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 22(2): 1138–1149.

Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 922–929.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, Z.; Chow, C.-Y.; and Zhang, J.-D. 2018. STANN: A spatio–temporal attentive neural network for traffic prediction. *IEEE Access*, 7: 4795–4806.

Huang, C.-Z. A.; Vaswani, A.; Uszkoreit, J.; Shazeer, N.; Simon, I.; Hawthorne, C.; Dai, A. M.; Hoffman, M. D.; Dinculescu, M.; and Eck, D. 2018. Music transformer. *arXiv preprint arXiv:1809.04281*.

Huang, W.; Song, G.; Hong, H.; and Xie, K. 2014. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5): 2191–2201.

Jabbarpour, M. R.; Zarrabi, H.; Khokhar, R. H.; Shamshirband, S.; and Choo, K. R. 2018. Applications of computational intelligence in vehicle traffic congestion problem: a survey. *Soft Comput.*, 22(7): 2299–2320.

Jin, D.; Huo, C.; Liang, C.; and Yang, L. 2021a. Heterogeneous graph neural network via attribute completion. In *Proceedings of the Web Conference 2021*, 391–400.

Jin, D.; Yu, Z.; Jiao, P.; Pan, S.; He, D.; Wu, J.; Yu, P.; and Zhang, W. 2021b. A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Transactions on Knowledge and Data Engineering*.

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, 95–104.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Liu, L.; Chen, J.; Wu, H.; Zhen, J.; Li, G.; and Lin, L. 2020. Physical-virtual collaboration modeling for intra-and inter-station metro ridership prediction. *IEEE Transactions on Intelligent Transportation Systems*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022.

Oord, A. v. d.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; and Kavukcuoglu, K. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Salinas, D.; Flunkert, V.; Gasthaus, J.; and Januschowski, T. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191.

Shih, S.-Y.; Sun, F.-K.; and Lee, H.-y. 2019. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8): 1421–1441.

Song, C.; Lin, Y.; Guo, S.; and Wan, H. 2020. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 914–921.

Stoller, D.; Tian, M.; Ewert, S.; and Dixon, S. 2019. Seq-u-net: A one-dimensional causal u-net for efficient sequence modelling. *arXiv preprint arXiv:1911.06393*.

Tsai, Y.-H. H.; Bai, S.; Yamada, M.; Morency, L.-P.; and Salakhutdinov, R. 2019. Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel. *arXiv preprint arXiv:1908.11775*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34: 22419–22430.

Wu, S.; Xiao, X.; Ding, Q.; Zhao, P.; Wei, Y.; and Huang, J. 2020a. Adversarial sparse transformer for time series forecasting. *Advances in neural information processing systems*, 33: 17105–17115.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020b. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*.

Xu, M.; Dai, W.; Liu, C.; Gao, X.; Lin, W.; Qi, G.-J.; and Xiong, H. 2020. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*.

Xu, M.; and Liu, H. 2021. Road Travel Time Prediction Based on Improved Graph Convolutional Network. *Mobile Information Systems*, 2021.

Yang, Y.; and Xu, Z. 2020. Rethinking the value of labels for improving class-imbalanced learning. *Advances in neural information processing systems*, 33: 19290–19301.

Yao, H.; Tang, X.; Wei, H.; Zheng, G.; and Li, Z. 2019. Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction. In *AAAI*, 5668–5675. AAAI Press.

Yu, B.; Yin, H.; and Zhu, Z. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Yu, B.; Yin, H.; and Zhu, Z. 2019. St-unet: A spatio-temporal u-network for graph-structured time series modeling. *arXiv preprint arXiv:1903.05631*.

Yu, Z.; Jin, D.; Liu, Z.; He, D.; Wang, X.; Tong, H.; and Han, J. 2021. AS-GCN: Adaptive semantic architecture of graph convolutional networks for text-rich networks. In *2021 IEEE International Conference on Data Mining (ICDM)*, 837–846. IEEE.

Zhang, J.; Shi, X.; Xie, J.; Ma, H.; King, I.; and Yeung, D.-Y. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*.

Zhang, J.; Zheng, Y.; and Qi, D. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Thirty-first AAAI conference on artificial intelligence*.

Zhang, Y.; Kang, B.; Hooi, B.; Yan, S.; and Feng, J. 2021. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*.

Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; and Li, H. 2020. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Trans. Intell. Transp. Syst.*, 21(9): 3848–3858.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11106–11115.

Zhou, Z.; Wang, Y.; Xie, X.; Chen, L.; and Liu, H. 2020a. RiskOracle: a minute-level citywide traffic accident forecasting framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1258–1265.

Zhou, Z.; Wang, Y.; Xie, X.; Chen, L.; and Zhu, C. 2020b. Foresee urban sparse traffic accidents: A spatiotemporal multi-granularity perspective. *IEEE Transactions on Knowledge and Data Engineering*.