Expert Systems With Applications 218 (2023) 119587

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# RPConvformer: A novel Transformer-based deep neural networks for traffic flow prediction☆

Yanjie Wen [a], Ping Xu [a,*], Zhihong Li [b], Wangtu Xu [c], Xiaoyu Wang [b]

[a] Central south university, Changsha, Hunan 410083, China
[b] Beijing University of Civil Engineering and Architecture, Beijing 102627, China
[c] Xiamen University, Xiamen Fujian, 361005, China

ARTICLE INFO

*Keywords:*
Intelligent transportation system
Traffic prediction
Transformer
Positional embedding
1D causal convolutional
Multi-head attention

ABSTRACT

Traffic prediction problem is one of the essential tasks of intelligent transportation system (ITS), alleviating traffic congestion effectively and promoting the intelligent development of urban traffic. To accommodate long-range dependencies, Transformer-based methods have been used in traffic prediction tasks due to the parallelizable processing of sequences and explanation of attention matrices compared with recurrent neural units (RNNs). However, the Transformer-based model has two limitations, on the one hand, it ignores the local correlation in the traffic state in its parallel processing of the sequence, on the other hand, the absolute positional embedding is adopted to represent the positional relationship of time nodes is destroyed when it comes to calculate attention score. To address two embarrassing shortcomings, a novel framework called RPConvformer is proposed, where the improved parts are 1D causal convolutional sequence embedding and relative position encoding. In sequence embedding, we develop a sequence embedding layer composed of convolutional units, which consist of origin 1D convolutional and 1D causal convolutional. The size of the receptive field of the convolution can focus on the local region correlation of the sequence. In relative position encoding, we introduce a bias vector to automatically learn the relative position information of time nodes when linearly mapping the feature tensor. We respect the encoding and decoding framework of the Transformer, the encoder is responsible for extracting historical traffic state information, and the decoder autoregressively predicts the future traffic state. The multi-head attention mechanism is adopted by both encoder and decoder aims to focus on rich temporal feature patterns. Moreover, key mask technique is used after computing attention matrix to mask the traffic state at missing moments improving the resilience of the model. Extensive experiments on two real-world traffic flow datasets. The results show that RPConvformer achieves the best performance compared to state-of-the-art time series models. Ablation experiments show that considering the local correlation of time series has a higher gain on prediction performance. Random mask experiments show that the model is robust when the historical data is less than 10% missing. In addition, multi-head attention matrix provides further explanation for the dependence between time nodes. RPConvformer as an improved Transformer-based model can provide new ideas for molding temporal dimension in traffic prediction tasks. Our code has been open-sourced at (https://github.com/YanJieWen/RPConvformer).

## 1. Introduction

Traffic flow prediction is one of the core issues in an Intelligent transportation system (ITS) (Humayun, Almufareh & Jhanjhi, 2022). According to the limited historical flow information of the links, accurately and efficiently predicting the flow of the road section in the future can alleviate traffic congestion and improve traffic efficiency and safety.

Traffic flow prediction refers to obtaining useful information from history through technical means and then outputting the most likely links flow in the future. In essence, the traffic flow prediction problem belongs to the classical time series modeling problem (Drakulic & Andreoli, 2022). Traditional traffic flow prediction methods, such as OD-based links back-stepping (Fisk, 1988), Traffic allocation algorithms based on optimization model (Zheng, Lin, Feng, & Chen, 2020). The disadvantage of these methods is that a number of information are derived

☆ This document is the results of the research project funded by the National Social Science Fund Project of China (21FGLB014).
* Corresponding author.
*E-mail addresses:* obitowen@csu.edu.cn (Y. Wen), xuping@csu.edu.cn (P. Xu), lizhihong@bucea.edu.cn (Z. Li), ato1981@xmu.edu.cn (W. Xu), 591176637@qq.com (X. Wang).
*URL:* https://github.com/YanJieWen (Y. Wen).

https://doi.org/10.1016/j.eswa.2023.119587
Received 16 September 2022; Received in revised form 1 January 2023; Accepted 18 January 2023
Available online 31 January 2023

from a small amount of data. Moreover, the computational performance of these methods cannot meet the real-time operations (Jin et al., 2021). Hence, with the explosive growth of multi-source data, the modeling ideas based on data-driven are widely discussed, mainly divided into statistical modeling and machine learning modeling. The statistical modeling mainly depends on calculating the historical time series features like average value, variance, autocorrelation coefficient (ACF), partial autocorrelation coefficient (PACF), etc. The related methods such as the historical average (HA) (Kaysi, Ben-Akiva, & Koutsopoulos, 1993), autoregressive moving average (ARIMA) (Isufi, Loukas, Simonetto, & Leus, 2016), and autoregressive conditional heteroskedasticity (ARCH) (Hamilton & Susmel, 1994). However, the simple statistical methods cannot accurately simulate the complex state of the road, and machine learning fit the time series curve by complex nonlinear functions are widely used, such as the support vector regression (SVR) (Castro-Neto, Jeong, Jeong, & Han, 2009) and $K-\text{nearest}$ neighbor (KNN) (Cheng, Lu, Peng, & Wu, 2018). Machine learning models have outperformed statistical models because of the capabilities of handling high-dimensional and complicated temporal data, but simple machine learning for traffic prediction has not been fully utilized until the rise of the deep neural networks (DNNs), as so-called deep learning (Ma, Tao, Wang, Yu, & Wang, 2015).

Deep learning-based models have been often used in solving traffic prediction problems by building rich neurons. The basic neurons including convolutional neural networks (CNNs) (Jiang, 2022) and recurrent neural networks (RNNs) (Van Lint, Hoogendoorn, & van Zuylen, 2002) have achieved good results in extracting traffic temporal features. CNNs have been proved to be able to adequately capture local features in the field of image processing, and also have good transferability on sequence datasets (Tang et al., 2020). Wu, Pan, Long, Jiang, and Zhang (2019) capture long-term sequence dependencies with stacked dilated 1D convolutional components. Ma et al. (2017) regards the traffic network as a image, and employees CNN to predict the traffic speed in the road network and evaluate the degree of congestion. Identically, RNNs can also better capture temporal features, but the deeper RNNs may cause shallow information failure, so the gate-based models are proposed to avoid the disappearance of gradient information. Sun, Boukerche, and Tao (2020) developed a SSGRU by stacking the gated recurrent unit (GRU) to mine temporal information. Tian, Zhang, Li, Lin, and Yang (2018) used long short-term memory unit (LSTM) to infer missing traffic states. Zheng et al. (2020) proposed a bidirectional LSTM (Bi-LSTM) module to extract the periodic features of traffic flow, thereby capturing the full variance trend of traffic flow, the Bi-LSTM considers the future information to realize context judgment. Ma et al. (2020) proposed a novel improved LSTM-based units, called NLSTM, which has stronger temporal representation ability than the stacked structure. The common feature of them is that the model set several types of gate structure to determine which information should be updated and which should be forgotten. In addition, researchers have paid more attention to modeling the spatial–temporal relationship of traffic states. Vijayalakshmi et al. (2021) developed an attention-based CNN-LSTM to make full use of details of spatial and temporal traffic data. Bogaerts, Masegosa, Angarita-Zapata, Onieva, and Hellinckx (2020) used graph convolution to model spatial features and LSTM to extract temporal features. Do, Vu, Vo, Liu, and Phung (2019) trained the GRU-CNN unit for spatial–temporal feature extraction, and the architectures of Seq2Seq can perform multi-step traffic flow prediction. Furthermore, to capture the spatial topological relationships of irregular grids, graph neural networks (GNNs) are used for spatial modeling (Jiang & Luo, 2022). In general, the core idea is CNNs or GNNs are used for spatial modeling and RNNs or their variants are exploited to extract temporal features. These units are connected by edges (like stacked or skipped He, Zhang, Ren, & Sun, 2016) to form a deep learning framework. However, both CNNs, GNNs and RNNs also have their limits on traffic prediction. **Firstly**, for the CNNs or GNNs, the convolution kernel only attends in a receptive field and graph convolutional only focus on the correlation with its nearest $K$ order neighbors, which ignores global correlation between the time nodes (Cordonnier, Loukas, & Jaggi, 2019). **Secondly**, for the RNNs, on the one hand, RNNs are recursive and cannot parallelize all sequences in parallel (Katharopoulos, Vyas, Pappas, & Fleuret, 2020), on the other hand, RNNs are weak for variable-length data (Reza, Ferreira, Machado, & Tavares, 2022).

Recently, the introduction of the attention mechanism (Vaswani et al., 2017) can solve the above problems very well. It can process all input data in parallel to ensure the calculation is global, and it can flexibly handle variable-length sequences through masking technology. In addition, the encoder–decoder (Seq2Seq) framework has been a universal framework employed in the field of computer vision (CV) and natural language processing (NLP) widely (Cho et al., 2014). The researchers put the attention mechanism into the encoder–decoder framework to form the Transformer (Vaswani et al., 2017) and carried out many experiments that proved to be a state-of-the-art deep learning framework. However, the vanilla Transformer was designed to solve the NLP problem firstly, which is obviously different from the traffic prediction. On the one hand, word processing is embedded by word index from the dictionary, which is obviously not applicable for traffic flow. On the other hand, the vanilla Transformer parallel deals with the input sequence, it cannot implicitly learn the position information of the sequence. It is necessary to add positional embedding into the vanilla Transformer. But vanilla Transformer adopts absolute position encoding, which only reflects the position relationship, and cannot distinguish the direction between sequences. Thus, it is a challenge work to transfer Transformer from NLP to traffic flow prediction.

In summary, we propose a novel Transformer framework with relative positional encoding and 1D convolutional layer improvements, called RPConvformer. First, in order to capture the relations of traffic flow, the word embedding layer was ditched and the 1D CNNs-based layer was adopted as traffic sequence embedding layer. The developed layers are not only to capture the local relations of sequences, but learn the causality of sequential sequences plausibly. The layers realize feature encoding of the input traffic flow tensor better processing by subsequent layers.

Second, considering the positional encoding ability of vanilla Transformer is weak, we introduce the relative position vector when linearly mapping the input tensor before calculating the attention score matrix, so as to improve the model's understanding of the sequence. We conducted extensive experiments on two public datasets (PEMS04 and PEMSD8), to demonstrate the effectiveness of our method in traffic flow forecasting compared with state-of-the-art baseline models. The main contributions of the current study are summarized as follows:

- We develop a novel Transformer-based improved model for traffic flow prediction, which preserves the characteristics of Transformer, such as encoder–decoder framework, multi-head attention, and key mask methods. Encoder–decoder is used to encode historical traffic status and predict future traffic status. Multi-head attention is responsible for focusing on the features patterns of sequences in different sub-spaces. Key mask endows the model flexibility and robustness, it can guarantee a certain prediction accuracy even in the absence of historical data.
- In order to fully exploit the potential of Transformer in traffic flow prediction, we discard the word embedding layer and develop a fully convolution embedding layer to learn local correlations and causality in traffic sequences. In addition, the relative positional encoding is employed for the linear mapping of the multi-head attention, and the relative position relationship between sequence nodes is obtained in a data-driven manner.
- Two public datasets are used to demonstrate the state-of-the-art of our model. The experimental results show that the full convolution as an embedding layer can significantly improve the prediction performance, key mask technology can enhance the flexibility and robustness of the model, and the multi-head attention mechanism can provide rich temporal feature patterns.

This paper is organized according to the following structures. Section 2 discussed the application of the deep learning framework in traffic flow prediction and Transformer with its improvement methods. Section 3 described a novel RPConvformer model. Section 4 included information about the experimental setups, results of experiments, and discussion. Section 5 draw the conclusions and future works.

## 2. Related work

### 2.1. Deep learning framework for traffic flow prediction

In previous studies on traffic flow forecasting, statistical time-series models or simple machine learning models have often been used. ARIMA and its variants typically forecast extremely short sections (Williams, 2001; Williams & Hoel, 2003), which encountered problems when a large number of data were involved, and simple statistical models cannot fit huge and complex data. Subsequently, models that utilize both ARIMA and Kalman filter (KF) to forecast traffic flow in different time periods (Stathopoulos & Karlaftis, 2003). These models typically overestimate when large and high-dimensional of data are involved (Miglani & Kumar, 2019).

As we all known, deep learning has a wide range of applications due to its complex neuron components and connection methods (Humayun, Ashfaq, Jhanjhi & Alsadun, 2022). To overcome the shortcoming of traditional traffic flow prediction models, deep learning-based have been adopted to operate traffic prediction tasks. Researchers utilize CNNs, RNNs, or their hybrids to compose a deep learning framework. CNNs can play two main roles in traffic flow prediction, on the one hand, CNNs can capture spatial dependencies, the traffic condition relationship of adjacent roads can be captured by the gridding road network (Zhang, Yu, Qi, Shu, & Wang, 2019; Zheng, Yang, Liu, Dai, & Zhang, 2019), on the other hand, CNNs also are able to focus on temporal correlations by using 1D convolution (1DConv) (Liu, Zheng, Feng, & Chen, 2017; Qiao, Wang, Ma, & Yang, 2021). 1DConv's filter has the shape $K \times C$, where $K$ is the kernel size, $C$ is the number of features channels. Besides 1DConv-based models are used to obtain temporal features, LSTMs or GRUs have been investigated for long term time series modeling. Ma et al. (2015) used remote microwave sensor data to train LSTMs to predict traffic speed. Shu, Cai, and Xiong (2021) proposed an improved bidirectional GRU (Bi-GRU) model to predict short-term traffic flow, and upgrade the optimizer and learning rate during the training phase to further improve the prediction accuracy of the model. Cui, Ke, Pu, and Wang (2018) developed a deep stacked bidirectional and unidirectional LSTM (SBU-LSTM) neural network architecture, which considers both forward and backward dependencies in time series data. Ma, Dai, and Zhou (2021) combined LSTMs and bidirectional LSTMs to overcome the large prediction errors for short-term traffic flow. At present, most studies on traffic prediction mainly focus on modeling spatial–temporal data. Capsule Network (a CNNs framework) and NLSTM (nested LSTMs) (Ma et al., 2020), graph convolution and CNNs (Yu, Yin, & Zhu, 2017), Wavenets (Chen, Song, & Zhao, 2021), ConvLSTM (Zhang, Lin, Li, & Wang, 2021). Although deep learning models have been widely studied, the above methods have limitations, such as parallel globalization of sequences and the problem of variable-length sequences.

Therefore, we propose a multi-head attention mechanism that can globally parallelize sequences, and novel masking techniques allow sequences to be elastically variable in length. In addition, the mainstream encoder–decoder framework is employed for training data. Transformer-based models are used for traffic flow prediction tasks.

### 2.2. Vanilla transformer and its variants

Vanilla transformer has achieved great achieved success in the field of NLP (Vaswani et al., 2017). Therefore, it is a natural idea to migrate the Transformer into the traffic domain for traffic flow prediction. According to the paper (Li et al., 2019), there are 4 points of view to believe that Transformer is suitable for traffic flow prediction: (1)Transformer can process input sequences in parallel, and the model is more efficient.(2) Transformer has the ability to rely on modeling for a longer period of time and is more effective in a long sequence, which RNN and its variants cannot completely eliminate the problems of gradient disappearance and explosion. (3) Transformer adopts multi-head attention which can focus on different patterns. (4) Explanation is all you need. By visualization, the attention score can view the distribution of the attention score about self-attention and interactive-attention output. Nevertheless, vanilla Transformer for traffic flow prediction also has disadvantages that the inapplicability of sequence embedding and weak positional embedding. Thus, some excellent improved strategies for sequence embedding and position embedding are proposed.

**Sequence embedding:** Vanilla Transformer is designed for NLP firstly. Before operating the multi-head attention mechanism, the sequence should be processed by word embedding layer, which is not suited for traffic flow sequence. In addition, the vanilla Transformer's self-attention calculation method is sensitiveness to local information, making the model vulnerable to outliers and bringing potential optimization problems. Correspondingly, Li et al. (2019) proposed a ConvTrans for time-series forecasting, which CNNs is employed to enhance local information. Zhou et al. (2021) adopted 1DConv for sequence embedding, which is a part of the Informer. In short, CNNs can be involved to in Transformer for local context modeling and bring encouraging results.

**Positional embedding:** Due to the Transformer processes sequence in parallel. In essence, it does not contain the sequential relationship between sequences. Thus, it is necessary to couple position information before implementing the attention mechanism. The improvement based on position embedding can be divided into absolute position embedding and relative position embedding. As for absolute position, Vaswani et al. (2017) proposed a sinusoidal positional embedding, which is the positional embedding method in vanilla Transformer. Others absolute embedding include training (Devlin, Chang, Lee, & Toutanova, 2018), recursion (Liu, Yu, Dhillon, & Hsieh, 2020). Although absolute positional embedding adds the information of the positional sequence, the direction between contexts is not fully considered. According to the paper (Yan, Deng, Li, & Qiu, 2019), when it comes to get an attention score, the positional embedding may be invalid. Thus the improved method is to introduce relative positional embedding into the multi-attention layer. Shaw, Uszkoreit, and Vaswani (2018) added a relative position offset vector in the multi-head attention mechanism module of the vanilla Transformer, which is a trainable vector aiming to let the model capture the relative position of different words in the same sequence automatically. Based on this idea, researchers have proposed varied Transformer-based variants framework, such as Transformer-XL (Dai et al., 2019), T5 model (Raffel et al., 2020), and DeBERTa (He, Liu, Gao, & Chen, 2020).

In view of the limitations of LSTMs or GRUs in time series modeling, Transformer was used to model traffic flow time series data. In addition, 1D convolutional network mapped input traffic flow sequence to capture local correlation and introduced relative positional embedding into the multi-head attention mechanism.

## 3. Methodology

This section presents firstly clarifies the problem definition to make the scope of our research clearly, then analyzes the shortcomings of the vanilla Transformer in traffic flow prediction, and finally introduces our proposed RPConvformer in detail.

### 3.1. Problem definition

In this paper, our main goal is to propose a high-performance deep learning-based model specialized for time-series forecasting and evaluate its applicability in traffic flow forecasting.

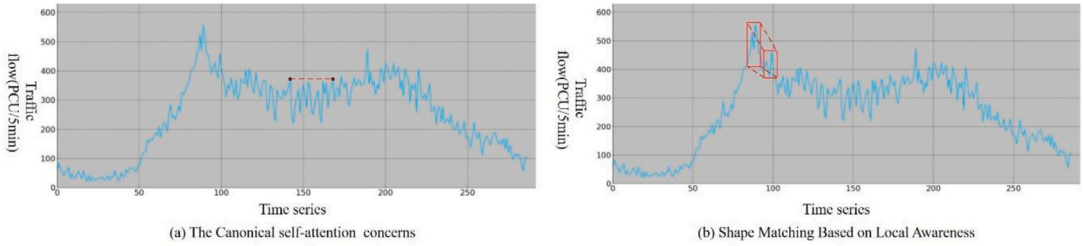


**Fig. 1.** The visualization results of PEMS04 on a working day. Because of attention score only shows the degree of correlation between single points of time as shown in (a), the middle red dot only focuses on another single time red dot that is close equal to the value, without considering its surrounding context. In contrast, we hope to enhance the ability of the model to build local sequences as shown in (b), that is, the trend changes will be considered when fitting values.

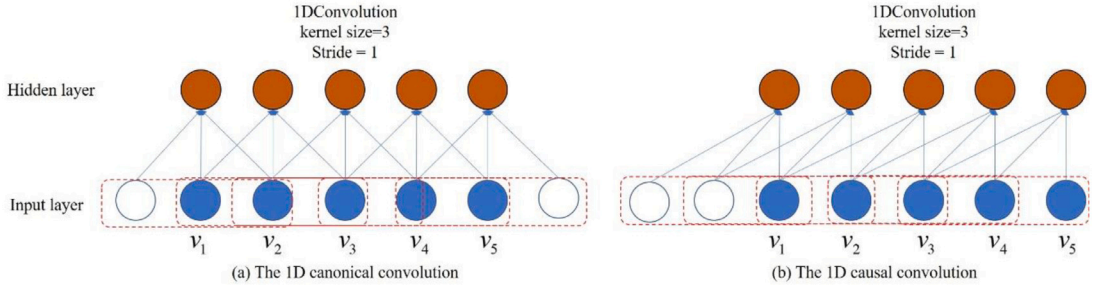


**Fig. 2.** The comparison between 1D canonical convolution (left) and 1D causal convolution (right). The blue solid is the traffic flow input at the corresponding time, the hollow ball is part of the padding, the orange solid represents the hidden layer information, and the red dashed box is 1D convolution sliding process with direction of from left to right.

Suppose we have a historical collection of $H$ related univariate time series $V_H \in \mathbb{R}^H$, where $V_H = \{v_1, \ldots, v_h, \ldots, v_H\}$ and $V_h$ denotes the number of traffic flow at time $h$. Our purpose is to fit a complex function to predict all traffic flow values $V_F = \{v_{H+1}, \ldots, v_f, \ldots, v_{H+F} \mid f > H\}$ at next $F$ time steps in a data-driven manner. Moreover, we reduce the traffic flow prediction problem to learning a multi-step-ahead forecasting model with the following form:

$$\left[\widehat{v_{H+1}}, \ldots, \widehat{v_{H+F}}\right] = f\left(\left[v_1, \ldots, v_H\right]; \theta\right) \tag{1}$$

Where $\widehat{v_f}$ represents the prediction results from model at time $f$, $\theta$ denotes the learnable parameters shared by all-time series in the collection. $\theta$ is continuously optimized by the loss function calculating by the ground truth and prediction, which is formulated as follows:

$$\theta^* = \arg\min_{\theta^*} \mathrm{Loss}\left(\widehat{V_F}, V_F; \theta^*\right) \tag{2}$$

Where $\theta^*$ is the optimal parameter for model fitting, $\widehat{V_F}$ is the predicted tensor, $V_F$ is the ground truth, and $\mathrm{Loss}(\cdot)$ is the loss function, and the mean square error (MSE) is generally employed as convergence evaluation index for the prediction regression task.

### *3.2. The limitation of vanilla transformer for traffic flow prediction and improvement strategies*

In this section, the shortcomings of the vanilla Transformer are analyzed for traffic flow prediction. There are two main points: **(1)** Word embedding in Vanilla transformer is adopted to solve NLP, which is not suited for traffic flow prediction on account of complex nonlinear characteristics. **(2)** Sinusoidal positional embedding in vanilla Transformer is used for positional embedding. When calculating the attention score, the positional information will be invalid. In this section, corresponding improvement strategies will be put forward for the above problems.

**The improvement of sequences embedding:** Due to various events, such as holidays and extreme weather, patterns in traffic flow may evolve with time significantly. Therefore, both outliers and trend change points are closely related to the surrounding context environment (Li et al., 2019). However, in vanilla Transformer, the attention score is calculated by point-wise inputs, which results in nodes with similar values to receive more attention and wrongly match as shown in Fig. 1(a). In contrast, we expect the model to correctly match the most relevant features according to the local shape of the sequence variation, as shown in 1(b). In order to ease the issue in vanilla Transformer, a CNNs-based framework is proposed to alter word embedding. On the one hand, CNNs is adapted to the feature extraction of traffic flow sequences. On the other hand, more abundant local information is obtained by setting up convolution kernels with different size.

1D convolution units are mainly used to handle time-series data and capture the local context in sequence. As shown in Fig. 2, it can be seen that the 1D canonical convolution set up a convolution kernel with a size equal to 3, and a stride value is 1 to extract sequence features. To pay more attention to local information, the kernel size is greater than 1 (Note that when kernel size is 1, the convolutional layer will degrade to a dense layer). In addition, to ensure that the input and output sequences are of the same length, appropriate padding is adopted. However, 1D canonical convolution padding is to supplement the two ends of the sequence, which will lead to the current position access to future information (see Fig. 2(a)). So, in order to avoid future disclosure of information, 1D causal convolution is suggested for sequence embedding, as shown in Fig. 2(b), it can be seen that padding is a one-end complement with the same output sequence, but the current embedding information is only related to itself and historical local input, which guarantees the sequential logic of the sequence.

Finally, we combine canonical convolutions and causal convolutions to construct a pure convolutional network as sequence embedding layer, so that the improved Transformer can be applied to the task of traffic flow prediction. The proposed sequence embedding layer is shown in Fig. 3, we assume that the input is the traffic flow $V \in \mathbb{R}^{B \times T_H \times 1}$, where $B$ represents the batch size. The stacked 1D canonical convolutions (1D conv in short) with a kernel size of 1 have the ability to scale sequence features without changing sequence length, i.e. $V \in \mathbb{R}^{B \times T_H \times 1} \xrightarrow{\text{1Dconv}} V' \in \mathbb{R}^{B \times T_H \times D}$. Next, 4 parallel 1D causal convolutions (Causal 1D conv in short) are used to enhance the locality of the sequence. The feature coupling extracted by multiple causal convolutions can obtain rich representations.

**The improvement of positional embedding:** Vanilla Transformer uses a sinusoidal function for positional embedding (Vaswani et al.,

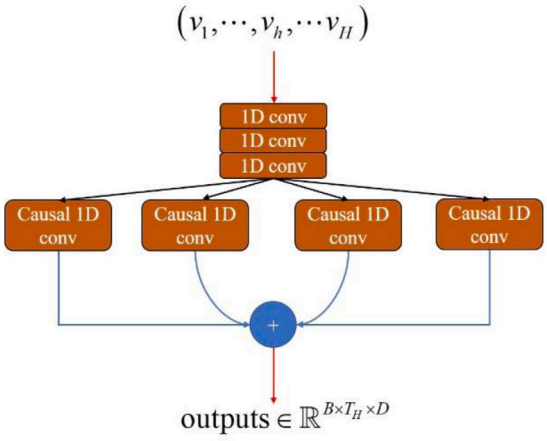


**Fig. 3.** The structure of sequence embedding with a pure 1D convolution network combined with 3 canonical convolutions with a kernel size of 1, and 4 1D causal convolutions with a kernel size of 9, according to the paper (Li et al., 2019).

2017), which is defined as:

$$\begin{cases} pos_{t,2d} = \sin\left({}^{t}/_{10000^{2d/D}}\right) \\ pos_{t,2d+1} = \cos\left({}^{t}/_{10000^{2d/D}}\right) \end{cases} \tag{3}$$

Where $t$ is the index of time series, $d$ is the index of traffic flow features after projection, and there are $D$ dimensions in total. For the positional embedding of traffic flow at time of $t$, the odd dimension adopts the cosine function, and the sine function is used to get an even dimension. The calculated positional vector is $pos_t = \left[\sin\left(c_0 t\right), \cos\left(c_0 t\right), \ldots, \sin\left(c_{d/2-1} t\right), \cos\left(c_{d/2-1} t\right)\right]$, the $c_i$ is the constant decided by $i$, and its value is equal to $1/10000^{2i/D}$, where $i \in \left[0, D/2-1\right]$.

The positional embedding based on sinusoidal is a fixed vector, not participating in the training. It is only used to provide a piece of positional information inside each sequence. Its visual result is as shown in Fig. 4. It is obvious that due to the nature of the sine/cosine function, each value of the positional vector lies in $[-1, 1]$. In addition, the right half of the figure is almost blue, this is result from $2d$ approaches to $D$, the smaller the frequency and the longer the wavelength. Therefore, the traffic condition at different time $t$ has little impact on the final result. In contrast, the more you go to the left, the more frequent the color alternation. Horizontally, for the position embedding at different times in the same sequence, they have different patterns. Therefore, the sinusoidal function can be used to represent the position information of the sequence. In spite of this, sinusoidal positional embedding may also fail when the information is passed to the multi-head attention mechanism layer. Next, the failure process of positional embedding will be discussed in detail. The general attention calculation process as so-called scaled Dot-Product attention (SDPA) as follows:

$$\begin{cases} q_i = \left(x_i + pos_i\right) W_q \\ k_j = \left(x_j + pos_j\right) W_k \\ v_j = \left(x_j + pos_j\right) W_v \\ a_{ij} = \mathrm{softmax}\left(\frac{q_i k_j^T}{\sqrt{D_k}}\right) \\ o_i = \sum_j a_{ij} v_j \end{cases} \tag{4}$$

Where $i, j$ refer to different time nodes, $x_i, x_j$ is the input of the attention layer with the shape $\mathbb{R}^D$, $pos_i \in \mathbb{R}^D$ is the positional embedding vector at the corresponding time, $q_i, k_j, v_j \in \mathbb{R}^D$ are quires,keys, and values, $W_q, W_k, W_v \in \mathbb{R}^{D \times D}$ are the weight matrix of quires, keys, and values respectively, which are trainable parameters in vanilla Transformer, softmax is the normalization function for row vectors, $D_k$ is the dimension number of keys, which is to scale attention score, $a_{ij}$ is the attention score between $i$ and $j$, which is the most important contents in attention layer, $o_i \in \mathbb{R}^D$ represents the output. To clarify the role of positional embedding in attention layer, we unfold $q_i k_j^T$ as follows:

$$\begin{aligned} & q_i k_j^T = \left(x_i + pos_i\right) W_q W_k^T \left(x_j + pos_j\right)^T \\ & = \underbrace{x_i W_q W_k^T x_j^T}_{a} + \underbrace{x_i W_q W_k^T pos_j^T}_{b} + \underbrace{pos_i W_q W_k^T x_j^T}_{c} + \underbrace{pos_i W_q W_k^T pos_j^T}_{d} \end{aligned} \tag{5}$$

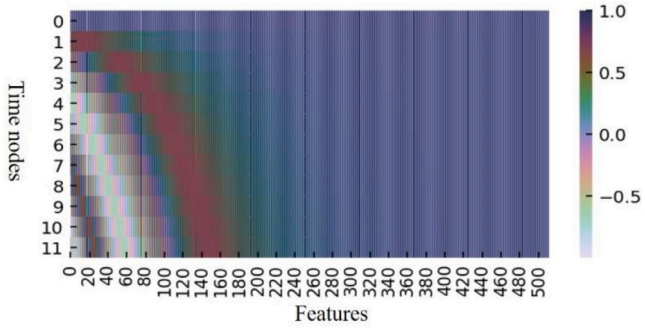


**Fig. 4.** The visualization of positional embedding. Assuming that the sequence length is 12, the features dimension of each time node is 512.

In Eq. (5), $a, b, c, d$ as the relationship between input and input, input and position, position and input, position and position respectively that only $d$ may contain the relative positional information of $i$ and $j$. We assume that $W_q W_k^T$ is an identity matrix, and let $j = i + k$, $k$ represents the time interval between $i$ and $j$. According to Eq. (3), the relative position relationship between $i$ and $j$ has the following form:

$$\begin{aligned} pos_i pos_{(i+k)}^T &= \sum_{z=0}^{d/2-1} \left[\sin\left(c_z t\right) \sin\left(c_z\left(t+k\right)\right) + \cos\left(c_z t\right) \cos\left(c_z\left(t+k\right)\right)\right] \\ &= \sum_{z=0}^{d/2-1} \cos\left(c_z k\right) \end{aligned} \tag{6}$$

In Eq. (6), ideally, the sinusoidal positional embedding contains the relative position information $k$. However, the paper (Yan et al., 2019) has proved a fact that the relative position will be destroyed after unknown linear changes ($W_q$ and $W_k$ are trainable random parameter matrices) through a large number of experiments. Therefore, the relative position information will be invalid when calculating the attention score.

In order to consider the relative position information of the sequence when calculating the attention score. Inspired by paper (Shaw et al., 2018), we introduce distance offset $b_{ij}^k$ and $b_{ij}^v$ to get keys and values respectively. These offsets are used to describe the distance $(i-j)$, which is a parameter that needs to be learned in the network. Finally, we introduce relative position information into Eq. (4), and the alternative SDPA is as follows:

$$\begin{cases} q_i = x_i W_q \\ k_j = x_j W_k + b_{ij}^k \\ v_j = x_j W_v + b_{ij}^v \\ a_{ij} = \mathrm{softmax}\left(\frac{q_i k_j^T}{\sqrt{D_k}}\right) \\ o_i = \sum_j a_{ij} v_j \end{cases} \tag{7}$$

Where $b_{ij}^k \in \mathbb{R}^D$ is the bias of keys, $b_{ij}^k in \mathbb{R}^D$ is the bias of values. These two variables are introduced into the SDPA (see Eq. (4)) to learn the relative positional relationship between temporal node $i$ and $j$. As for the other variables, the definitions conform to Eq. (4).

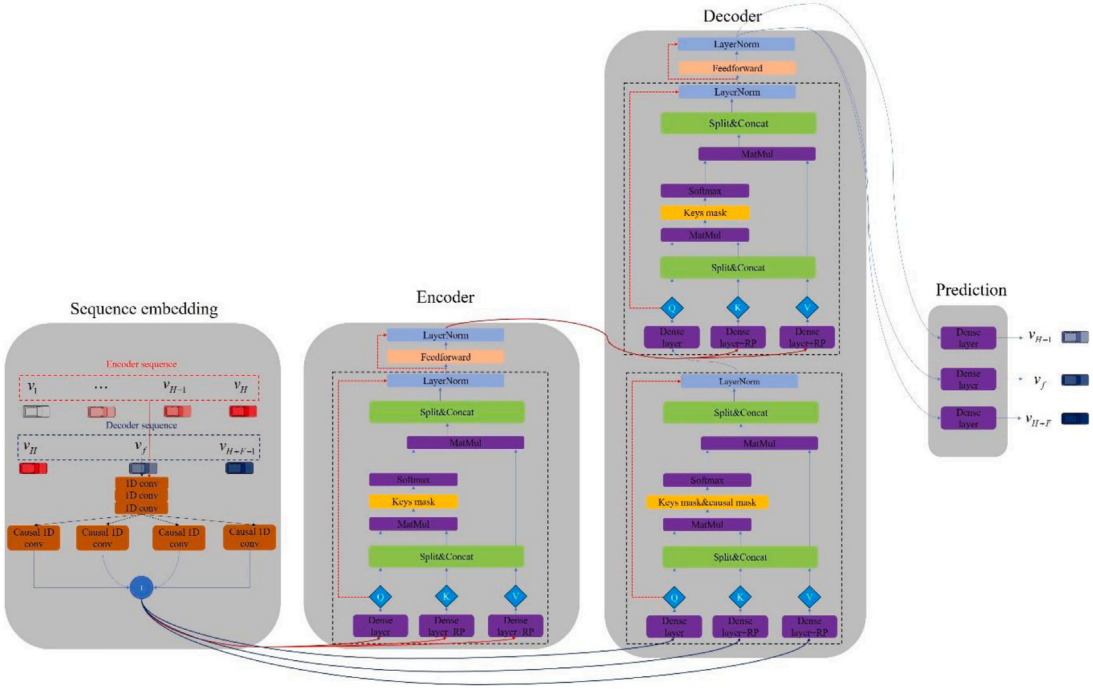


**Fig. 5.** The framework of RPConvformer. There are 4 blocks in the framework, purple boxes represents the calculation operations, different colored boxes represent different modules, the black dotted rectangle is a multi-head attention mechanism module, and the line is the flow direction of the tensor.

### 3.3. Framework of RPConvfomer

#### 3.3.1. Overall pipeline

To model the dynamics in traffic flow data, we propose a novel RPConvformer to be responsible for traffic flow prediction, which is an improvement on Transformer. Specifically, the improvements include a pure convolutional network is employed to build a sequence embedding layer to capture the local correlation of traffic states, and relative position information is introduced into attention layer giving a richer time node arrangement logic. The overall framework is shown in Fig. 5. The proposed model also has the encoder–decoder structure. Both the encoder and decoder stack multiple identical sub-layers, meanwhile, it is also necessary to guard against overfitting and memory overflow. In order to avoid the loss of information in the learning process, a skip connection is involved inside the encoder and decoder blocks. To adapt to the input of variable length sequences and the leakage of future information, we use the key mask and causal mask respectively. In general, the pipeline of the RPConvformer is described as follows:

1. The input of the RPConvformer is the historical traffic states $H \in \mathbb{R}^{B\times T_H\times 1}$ and zeros matrix with the same shape as output tensors $P \in \mathbb{R}^{B\times T_F\times 1}$. It should be noted that teaching forcing (Mihaylova & Martins, 2019) is adopted in the training phase. In the inference phase, future information cannot be used as input, so a slight change is required for $P$, that is, $P \leftarrow [H[:,-1:,:]\circ P[:,:-1,:]]$. Where $[\cdot\circ\cdot]$ represents concatenation. The sequence embedding projects them to $H^{(0)} \in \mathbb{R}^{B\times T_H\times D}$ and $P^{(0)} \in \mathbb{R}^{B\times T_F\times D}$. Where $B$ is the batch size, $T_H, T_F$ is the length of the historical sequence and the prediction sequence respectively, and $D$ represents the number of hidden units in RPConvformer.
2. The projected feature $H^{(0)}$ is fed into the encoder layer and $P^{(0)}$ is fed into the decoder layer, $H^{(1)}$ and $P^{(0)}$ are sent to the decoder to get the result $P^{(1)}$.
3. The decoder predicts future traffic flows in an autoregressive manner. That is $P^{(1)} \in \mathbb{R}^{B\times T_F\times D}$ is used to update $P^{(1)}$ in the time dimension. The above steps should be repeated $T_F$ times. Finally, the initial all zeros matrix $P$ is replaced by future traffic state step by step.

The details of the pipeline of the RPConvformer are shown in Algorithm 1.

**Algorithm 1:** Pipeline of RPConvformer

**Input**: $H \in \mathbb{R}^{B\times T_H\times 1}, P = 0, P \in \mathbb{R}^{B\times T_F\times 1}$
**Output**: Updated $P \in \mathbb{R}^{B\times T_F\times 1}$
$H^{(0)} \leftarrow$ Sequence embedding$(H)$;
$H^{(1)} \leftarrow$ Encoder$(H^{(0)})$;
**for** $t$ *in* $0 : T_F$ **do**
  $P \leftarrow [H[:,-1:,:]\circ P[:,:-1,:]]$;
  $P^{(0)} \leftarrow$ Sequence embedding$(P)$;
  $P^{(1)} \leftarrow$ Decoder$(H^{(1)}, P^{(0)})$;
  $P[:,t,:] \leftarrow P^{(1)}[:,t,:]$;
**end**
**Result**: $P$

#### 3.3.2. Encoder

The encoder layer receives the historical traffic flow feature tensor after sequence embedding layer (see Section 3.2) projection. The encoder layer consists of three components, namely multi-head attention (MMA), feed forward network (FFN), and layer normalization layer (LN). They are responsible for learning the features of historical sequences. The encoder layer can be stacked times, if necessary. In MMA, we introduce a bias into the linear projection to consider the relative position at different times nodes. FFN performs a position-wise transformation to realize tensor dimension changes. LN processes the tensor in batch normalization to accelerate the convergence (Ba, Kiros, & Hinton, 2016).

The core in encoder layer is MMA that computes feature tensors by using a modified scaled Dot-Product attention (see Eq. 7). Multi heads refers to cutting $queries, keys, values \in \mathbb{R}^{B\times T_H\times D}$ into $N$ sub-tensors, and each sub-tensor performs improved SDPA to focus on different aspects of sequence context semantics. Finally, the results of each head is spliced as the output of MMA. In addition, we also introduce the key mask method in the MMA, which can adapt to the input of any length sequence or incomplete sequence. Specifically, invalid time nodes in the

sequence will be filled with 0. The padded part in the attention matrix will be masked by maximal negative numbers to eliminate the influence of the padding part on the computational attention distribution.

The details of the key mask for encoder (KME) are shown in Algorithm 2. Where $Att$ is the multi-head attention score matrix before softmax normalization, $enc$ is the input after passing the sequence embedding block.

**Algorithm 2:** Key mask for encoder(KME)

**Input**: $Att \in \mathbb{R}^{(B\times N)\times T_H\times T_H}$,$enc \in \mathbb{R}^{B\times T_H\times D}$
**Output**: $Att \in \mathbb{R}^{(B\times N)\times T_H\times T_H}$,$mask\ memory \in \mathbb{R}^{B\times 1\times T_H}$

$key\ mask \leftarrow \text{expand\_dims}(\text{sign}(\sum_{d=1}^{D} |enc[:,:,d]|), \text{axis} = -1)$;
$key\ mask \leftarrow \text{transpose}(key\ mask, [0,2,1])$;
$mask\ memory \leftarrow key\ mask$;
$key\ mask \leftarrow \text{tile}(key\ mask, [N, T_H, 1])$;
$padding \leftarrow \text{ones}((B\times N), T_H, T_H)\times(-2^{32}+1)$;
$key\ mask\,[key\ mask == 0] \leftarrow padding$;
$key\ mask\,[key\ mask \neq 0] \leftarrow Att$;
$Att \leftarrow key\ mask$;
**Result**: $Att, mask\ memory$

FFN is composed of two layers of dense layers. The first layer adopts Leaky Relu nonlinear function mapping (Xu, Wang, Chen, & Li, 2015) and the second layer is the linear mapping. It has the following form:

$$FFN(x) = \text{LeakyRelu}\left(xW_1 + b_1\right)W_2 + b_2 \tag{8}$$

Where $x \in \mathbb{R}^{B\times T_H\times D}$ is the input of FFN, $W_1 \in \mathbb{R}^{D\times 4D}, b_1 \in \mathbb{R}^{4D}, W_2 \in \mathbb{R}^{4D\times D}, b_2 \in \mathbb{R}^{D}$ are trainable parameters. In addition, the skip connection (He et al., 2016) and layer normalization (LN) (Ba et al., 2016) are employed to enable better learning for deep networks.

The details of encoder layer are shown in Algorithm 3. Where $M$ is the number of encoder stacks, $N$ is the number of heads.

**Algorithm 3:** Encoder

**Input**: $enc \in \mathbb{R}^{B\times T_H\times D}$
**Output**: $enc\ memory \in \mathbb{R}^{B\times T_h\times D}, mask\ memory \in \mathbb{R}^{B\times 1\times T_H}$

$enc\_ \leftarrow enc$;
**for** $m$ *in* $0 : M$ **do**
  $Q \leftarrow enc\_\times W_q, W_Q \in \mathbb{R}^{D\times D}$;
  $K \leftarrow enc\_\times W_k + b_k, W_k \in \mathbb{R}^{D\times D}, b_k \in \mathbb{R}^{D}$;
  $V \leftarrow enc\_\times V_v + b_v, W_v \in \mathbb{R}^{D\times D}, b_v \in \mathbb{R}^{D}$;
  $Q\_ \leftarrow \text{concat}(\text{split}(Q, N, axis = -1), axis = 0)$;
  $K\_ \leftarrow \text{concat}(\text{split}(K, N, axis = -1), axis = 0)$;
  $V\_ \leftarrow \text{concat}(\text{split}(V, N, axis = -1), axis = 0)$;
  $Att \leftarrow \frac{Q\_\times(K\_)^{\mathrm{T}}}{\sqrt{D}}$;
  $Att, mask\ memory \leftarrow \text{KME}(Att, enc)$;
  $outputs \leftarrow \text{Softmax}(Att)$;
  $outputs \leftarrow outputs \times V\_$;
  $outputs \leftarrow \text{concat}(\text{split}(outputs, N, axis = 0), axis = -1)$;
  $outputs \leftarrow \text{LN}(outputs + enc)$;
  $outputs \leftarrow \text{LN}[\text{FFN}(outputs) + outputs]$;
  $enc\_ \leftarrow outputs$;
**end**
$enc\ memory \leftarrow outputs$;
**Result**: $enc\ memory, mask\ memory$

### 3.3.3. Decoder

The decoder layer contains two types of multi-head attention modules, which are the multi-head self-attention (MSA) and the multi-head interactive-attention (MIA) respectively. MSA is responsible for the attention distribution calculation of time nodes in the future sequence. MIA is in charge of computing the interactive attention distribution of historical sequence and future sequence.

The calculation of the attention distribution still conforms to Eq. (7). A teaching forcing strategy is adopted to correct the prediction results, i.e. feeding the decoder ground truth during the training stage. In short, the decoder layer receives 3 tensor, namely $dec \in \mathbb{R}^{B\times T_F\times D}$, $enc\ memory \in \mathbb{R}^{B\times T_F\times D}$, and $mask\ memory \in \mathbb{R}^{B\times 1\times T_H}$, where $dec$ is used as the input of MSA to perform teaching forcing which the future sequence right shifted and mapped by sequence embedding layer, $enc\ memory$ works as $k, v$ in MIA, so $mask\ memory$ is required as a tool of key mask.

In the MSA, the calculation process is very similar to Algorithm 3. Furthermore, a novel causal mask combined the key mask performs the masking work in MSA to prevent future information leakage, ensure more stable training. In other words, the traffic state of the next step is only related to the historical and current traffic state, which is a causal relationship obviously, so we call the masking method as causal mask in the decoder layer (CMD). The specific implementation details of CMD are shown in Algorithm 4.

**Algorithm 4:** Causal mask for decoder(CMD)

**Input**: $Att \in \mathbb{R}^{(B\times N)\times T_F\times T_F}$
**Output**: Updated $Att \in \mathbb{R}^{(B\times N)\times T_F\times T_F}$

$causal\ mask \leftarrow \text{ones}(T_F, T_F)$;
$causal\ mask \leftarrow \text{Lower Triangular}(causal\ mask)$;
$causal\ mask \leftarrow \text{tile}(\text{expand\_dims}(causal\ mask, 0), [(B\times N), 1, 1])$;
$padding \leftarrow \text{ones}((\text{B}\times\text{N}), \text{T}_\text{F}, \text{T}_\text{F})\times(-2^{32}+1)$;
$causal\ mask\,[causal\ mask == 0] \leftarrow padding$;
$causal\ mask\,[causal\ mask \neq 0] \leftarrow Att$;
$Att \leftarrow causal\ mask$;
**Result**: $Att$

In the MIA, the difference from Algorithm 3 is the interactive computation of the output of the encoder enc memory $\in \mathbb{R}^{B\times T_H\times D}$ and decoder dec memory $\in \mathbb{R}^{B\times T_F\times D}$, which is the output of MSA. Specifically, the key and value are then replaced by from the encoder, and query is from. The details of the MIA are shown in Algorithm 5.

**Algorithm 5:** Multi-head interactive-attention(MIA)

**Input**: $mask\ memory \in \mathbb{R}^{B\times 1\times T_H}$,$enc\ memory \in \mathbb{R}^{B\times T_H\times D}$,$dec\ memory \in \mathbb{R}^{B\times T_F\times D}$,$M$,$N$
**Output**: $outputs \in \mathbb{R}^{B\times T_F\times D}$

**for** $m$ *in* $0 : M$ **do**
  $Q \leftarrow dec\ memory \times W_q, W_q \in \mathbb{R}^{D\times D}$;
  $K \leftarrow enc\ memory \times W_k + b_k, W_k \in \mathbb{R}^{D\times D}, b_k \in \mathbb{R}^{D}$;
  $V \leftarrow enc\ memory \times W_v + b_v, W_v \in \mathbb{R}^{D\times D}, b_v \in \mathbb{R}^{D}$;
  $Q\_ \leftarrow \text{concat}(\text{split}(Q, N, axis = -1), axis = 0)$;
  $K\_ \leftarrow \text{concat}(\text{split}(K, N, axis = -1), axis = 0)$;
  $V\_ \leftarrow \text{concat}(\text{split}(V, N, axis = -1), axis = 0)$;
  $Att \leftarrow \frac{Q\_\times(K\_)^{\mathrm{T}}}{\sqrt{D}}$;
  $key\ mask \leftarrow \text{tile}\left(mask\ memory, \left[N, T_F, 1\right]\right)$;
  $padding \leftarrow \text{ones}\left(BN, T_F, T_H\right)\times\left(-2^{32}+1\right)$;
  $key\ mask\,[key\ mask == 0] \leftarrow padding$;
  $key\ mask\,[key\ mask \neq 0] \leftarrow Att$;
  $Att \leftarrow key\ mask$;
  $outputs \leftarrow \text{Softmax}(Att)$;
  $outputs \leftarrow outputs \times V\_$;
  $outputs \leftarrow \text{concat}(\text{split}(outputs, N, axis = 0), axis = -1)$;
  $outputs \leftarrow \text{LN}(\text{outputs} + \text{dec memory})$;
  $outputs \leftarrow \text{LN}[\text{FFN}(outputs) + outputs]$;
  $dec\ memory \leftarrow outputs$;
**end**
**Result**: $outputs$

Finally, the output of the decoder outputs $\in \mathbb{R}^{B\times T_F\times D}$ is passed through the output head with a linear dense layer and predict the value of traffic flow in the next $F$ steps. The specific process is as follows:

$$\hat{y} = dec\_memory \times W_{out} \tag{9}$$

**Table 1**
Datasets description.

| Datasets | The number of sensors | Interval | Time range | Sample points of time nodes |
|---|---|---|---|---|
| PEMS04 | 307 | 5 min | 1/1/2018–2/28/2018 | 16992 |
| PEMS08 | 170 | 5 min | 7/1/2016–8/31/2016 | 17856 |

**Table 2**
The division details of PEMS04 and PEMSD8.

| Datasets | Training days | Validation days | Testing days |
|---|---|---|---|
| PEMS04 | 44 | 5 | 10 |
| PEMSD8 | 47 | 5 | 10 |

Where $\hat{y} \in \mathbb{R}^{B \times T_F \times 1}$ is the prediction tensor of the model, $W_{out} \in \mathbb{R}^{D \times 1}$ is the linear transformation matrix.

## 4. Experiments and discussion

In this section, we conduct an extensive survey on the performance of RPConvformer on task of traffic flow forecasting, with two public datasets as benchmark for evaluation. It mainly includes the prediction performance comparison between RPConvformer and 7 baseline models, ablation experiments, random mask robustness evaluation and explanation.

### 4.1. Datasets and data analysis

**Datasets introduction:** In order to verify the validity of the RPConvformer, comparative experiments are conducted on two large, public datasets: PEMS04 and PEMSD8. The details information are summarized in Table 1. The datasets are collected by PEMS (Performance Measurement System) of Caltrans (California Department of Transportation) in highway traffic flow. The flow data is aggregated to 5 min so that there are 288 sample points a day (Chen, Petty, Skabardonis, Varaiya, & Jia, 2001).

**Data analysis:** To perform accurate and meaningful predictions, time-series data analysis involves an inherent understanding of the data, such as trends and seasonality. Generally, time-series data can be decomposed into the trend, the seasonal, and residuals, as shown in Fig. 6, we visualized 1K continuously changing sample points for PEMS04 and PEMS08. According to the decomposition results, it can be seen that the traffic flow data has certain regularity, the trend item is a stable trend similar to a sine function, the seasonal item shows a stable periodicity, and the residual item shows that PEMSD8 has higher stability than PEMS04 which has some protruding outliers. Although both datasets belong to the traffic flow data collected by highways, the time series features are slightly diversity that result from the influenced by the time range and regional differences. A clumsy method is to model the data separately.

Moreover, in order to objectively analyze the law of traffic flow sequence, the autocorrelation coefficient (ACF) and partial autocorrelation coefficient (PACF) are adopted to measure the regularity of the time series. ACF describes the autocorrelation between a time node and another, include direct and indirect information, and PACF only focus on the relationship between the current observation and its lag. As shown in Fig. 7, the datasets are trailed on ACF and truncated on PACF, which indicates that datasets follow small patterns and are predictable (Reza et al., 2022). In addition, Traffic flow data has a strong temporal correlation, and the traffic state closer to the current moment has the highest correlation coefficient. Therefore, it is a feasible scheme to infer future traffic flow based on historical traffic flow data.

Finally, due to the values of traffic flow data in different time periods have great differences. For example, the traffic flow in the morning peak and evening peak is very large, while the traffic flow in the early morning almost close 0 in most regions. This significant difference in traffic can mislead the model. Therefore, it is necessary to normalize input data before feeding into the model to ensure each data point has the same proportion. The advantage of this is that it can make the training process smoother and avoid training failure caused by large gradients. It is noted that in the testing phase, it is necessary to inverse normalize the data and obtain the original data for generalization performance evaluation. Specifically, the normalization calculation method is as follows:

$$v' \leftarrow \frac{v - \min(V)}{\max(V) - \min(V)} \tag{10}$$

Where $V$ is the traffic flow sequence, and $v$ is the flow at each time node.

### 4.2. Experimental settings

**Equipment description:** In order to ensure the comparability and repeatability of the experiments, all experiments are carried out in the deep learning framework Tensorflow-2.2 (It is reported that most transformers and their variants are built based on Pytorch). The hardware device is NVIDIA GeForce Rtx2080Ti 11G.

**Super-parameters definition:** Super-parameters refer to those that need to be manually set in advance before training phase. The optimal super-parameters are determined following several trails-error testing. The batch size $B$ is 32, the learning rate is 0.001, the number epochs is 100, the gradient update algorithm uses Adam (Kingma & Ba, 2014), the hidden units $D$ is 128, the number of staked encoder and decoder is 1, the number of attention head is 8, and the goal is to predict the traffic state for the next 12 time steps.

**The ratio of datasets partition:** In the training phase, the historical data is fed into the encoder and the right shifted future data is employed as decoder for teaching forcing to predict all the states in the next 12-time steps. In the testing stage, the results of prediction are performed autoregressively following Algorithm 1. We divide the data into training set, validation set, and testing set. The training set is used to update the network parameters, the validation set is used to monitor the indicators in the training process, and the testing set is used to evaluate the generalization performance of the model. The details of data division are shown in Table 2.

**Evaluation metrics**: In order to compare the generalization performance of each model, three metrics are adopted: the mean absolute error (MAE), the mean square error (MSE), and the mean absolute percentage error (MAPE). They are defined as follows:

$$\begin{cases} \text{MAE} = \frac{1}{N}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \\ \text{MSE} = \frac{1}{N}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \\ \text{MAPE} = \frac{1}{N}\sum_{i=1}^{n}\frac{\left|y_i - \hat{y}_i\right|}{y_i} \times 100\% \end{cases} \tag{11}$$

Where $y_i, \hat{y}_i \in \mathbb{R}^{T_F}$ represents the ground truth and the prediction value respectively, $N$ is the number of testing samples. Moreover, we focus on the prediction accuracy metrics with future time steps of 3, 6, 9 and 12 respectively.

### 4.3. Comparison and analysis

RPConvformer is compared with 7 excellent temporal modeling model on 2 datasets. These models can be divided into 3 different categories, the specific description of each model is shown in Table 3.

The performance of different methods for traffic flow prediction over the horizons of 3, 6, 9, and 12 is shown in Table 4. It can be seen that RPConvformer outperforms all the baseline methods. On the whole, deep learning methods have gained obvious advantages in modeling time series problems, which is benefit from the ability to fit the nonlinear and complex relationships of sequence datasets. On PEMS04, RPConvformer improves the second-best results in 3rd,

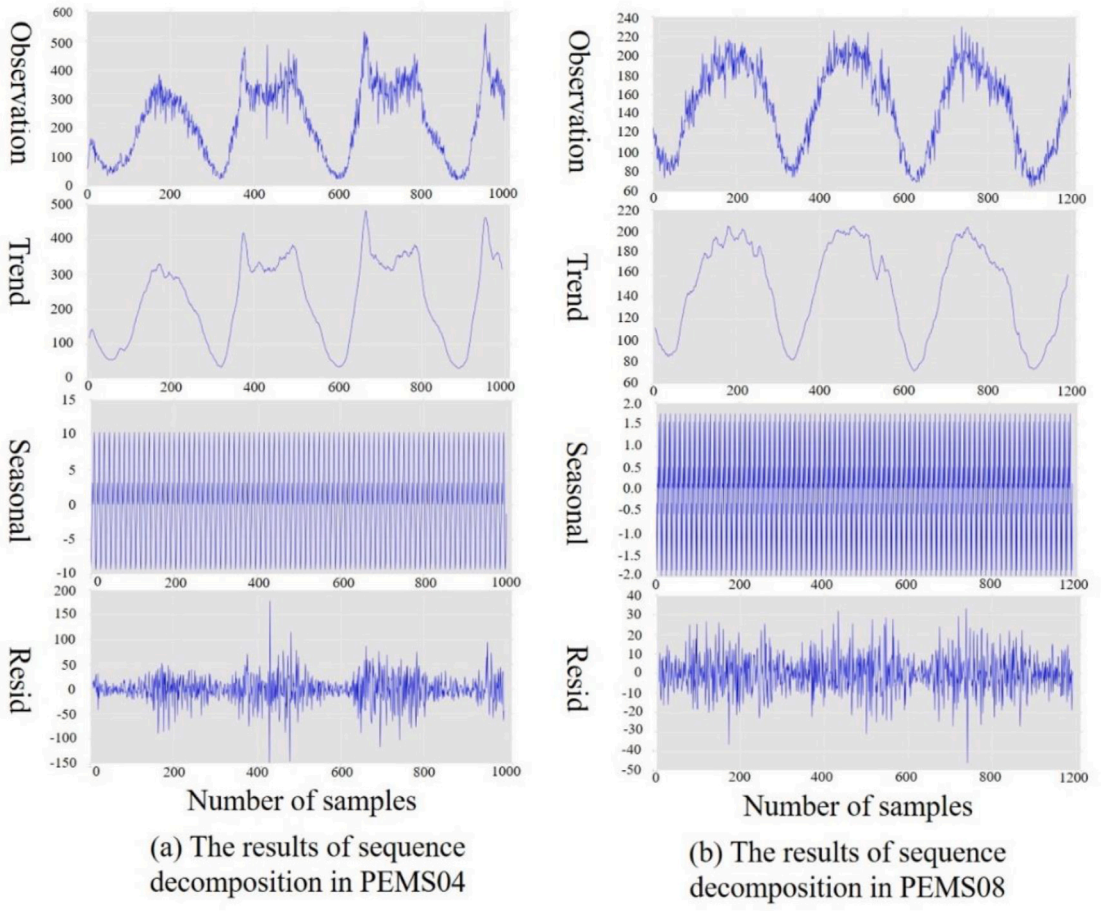


**Fig. 6.** Data visualization of 1K sample points in PEMS04 (left) and PEMSD8 (right). From the top to the bottom 1st, 2nd, 3rd, and 4th graphs represents the observation, trends, seasonality, and residual data respectively.

**Table 3**
The description of baseline models.

| Statistical method | Models | Description |
|---|---|---|
| Statistical method | Historical Average (HA) | The average value of historical and current traffic flow is involved as the prediction value for the next step. |
| | Autoregressive Integrated Moving Average model (ARIMA) (Williams & Hoel, 2003) | It includes moving average (MA) and autoregression (AR). The number of moving average terms $q$, autoregression terms $p$ and difference terms $d$ are 1, 8, and 0, respectively. |
| Simple machine learning | Support Vector Regression (SVR) (Castro-Neto et al., 2009) | This model is an extension of support vector machine classification (SVM) in regression problems. Radial basis function (RBF) as the kernel function |
| | K-Nearest Neighbor (KNN) (Cheng et al., 2018) | Giving a new sample $V$, find $k$ approximate samples in the training space $\mathbb{N}$ and calculate average as the prediction of the new samples. Setting $k = 5$ |
| Deep learning | Long Short-Term Memory (LSTM) (Tian et al., 2018) | It is a gated loop structure with 3 gates including update gate, forget gate and output gate. |
| | Gated Recurrent Unit (GRU) (Sun et al., 2020) | It is an improved version of LSTM includes reset gate and update gate. |
| | Bi-directional LSTM (BILSTM) (Cui et al., 2018) | It is a variant of LSTM. It learns the following information of the model by reversing the sequence, to consider the context information |

6th, 9th, 12th step by 9.92%, 17.05%, 22.93%, 26.16% in terms of MAE, 20.69%, 29.70%, 38.03%, 42.80% in terms of MSE, and 1.25%, 15.24%, 22.25%, 26.83%, in terms of MAPE. As for the average error, the performance has achieved 17.85%, 31.73%, 15.10% improvements. Similar improvement can be seen on PEMSD8, where the ratios of improvement in 3rd, 6th, 9th, 12th step are 5.12%, 12.46%, 18.17%, 23.09%, in terms of MAE, 12.37%, 20.46%, 29.93%, 37.83% in terms of MSE, and 3.64%, 13.15%, 16.18%, 19.61% in terms of MAPE. For the average error, RPConvformer higher than the second-best model than 13.56%, 24.50%, 12.52%. In summary, RPConvformer shows strong robustness in long-range time dependencies modeling, which thanks to parallel processing for each time node, while RNNs adopt recursive to fit the prediction results of each step leading to the gradual expansion of error.

Moreover, we visualize the comparison results of every step to clearly compare the performance differences of methods as shown in Fig. 8. The advantage of RPConvformer becomes more evident as time progress, indicating the competence of RPConvformer in longer-term prediction.

KNN and SVR considers recent traffic flow and constructs more complicated models have achieved greater success in traffic flow prediction compared with statistical method. However, simple machine

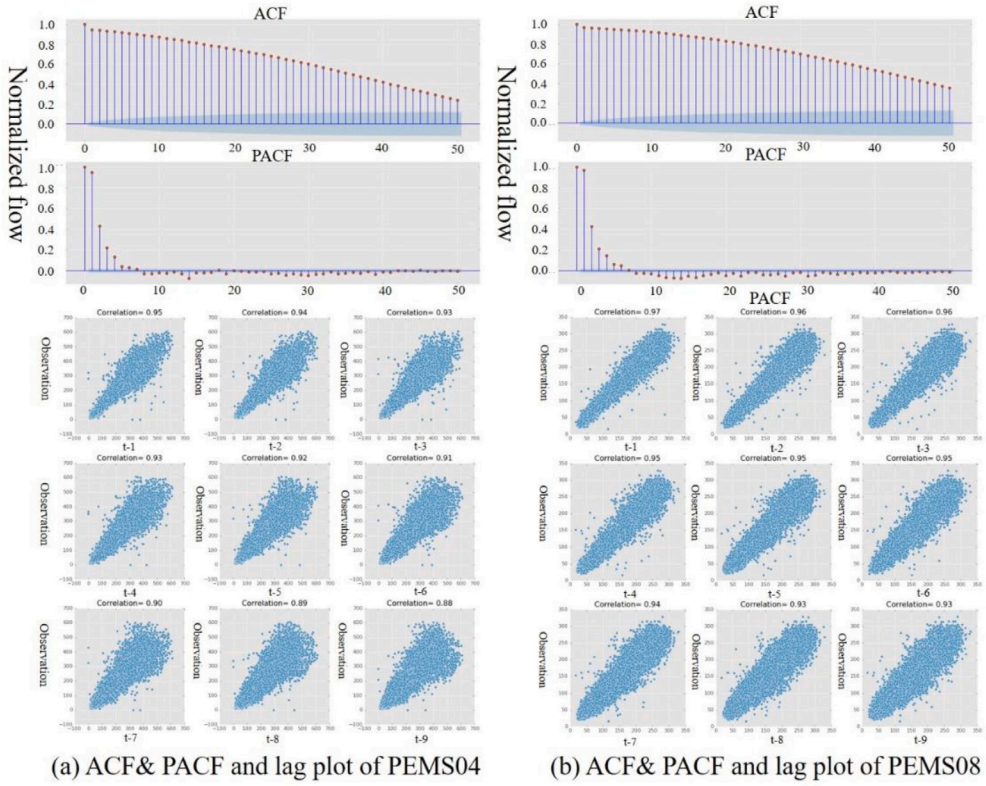


**Fig. 7.** Autocorrelation coefficient (ACF) and partial autocorrelation coefficient (PACF) analysis of the used dataset.: the upper two graphs represent ACF and PACF, the blue transparent area is the 95% confidence interval. The bottom graphs illustrate lag scatter analysis.

**Table 4**
Performance comparison of different methods with 3, 6, 9, 12 time steps on datasets PEMS04 and PEMSD8.

| Comparison on datasets of PEMS04(MAE/MSE($\times 10^3$)/MAPE(%)) | | | | | | |
|---|---|---|---|---|---|---|
| Type | Model | 3 steps | 6 steps | 9 steps | 12 steps | Average error |
| Statistical | HA | 49.17/5.10/24.90 | 46.55/4.78/22.89 | 44.67/4.60/21.64 | 47.17/4.92/22.98 | 47.13/4.87/23.35 |
| | ARIMA | 37.67/2.63/19.18 | 47.07/3.98/25.19 | 56.40/5.53/31.27 | 64.34/7.09/36.93 | 48.35/4.33/26.16 |
| Simple machine learning | SVR | 30.69/1.79/16.19 | 38.77/2.84/20.01 | 46.26/3.99/24.46 | 53.81/5.11/29.02 | 40.30/3.14/21.33 |
| | KNN | 29.73[a]/1.74/13.57[a] | 37.69/2.76/17.57 | 45.70/4.05/21.76 | 52.86/5.22/26.10 | 39.33/3.13/18.61 |
| Deep learning | LSTM | 29.82/1.74/14.00 | 32.61[a]/2.02[a]/15.81[a] | 35.41[a]/2.34[a]/17.17[a] | 38.50[a]/2.71[a]/19.23[a] | 32.94[a]/2.08[a]/15.96[a] |
| | GRU | 30.03/1.73[a]/14.43 | 33.50/2.06/17.66 | 36.40/2.38/19.66 | 39.78/2.73/22.52 | 33.56/2.09/17.53 |
| | BILSTM | 30.58/1.79/14.43 | 34.39/2.21/16.88 | 38.39/2.66/19.19 | 39.89/2.92/19.56 | 34.52/2.25/16.79 |
| | RPConvformer | **26.78/1.38/13.40** | **27.05/1.42/13.40** | **27.29/1.45/13.35** | **28.42/1.55/14.07** | **27.06/1.42/13.55** |
| Comparison on datasets of PEMSD8(MAE/MSE($\times 10^2$)/MAPE(%)) | | | | | | |
| Type | Model | 3 steps | 6 steps | 9 steps | 12 steps | Average error |
| Statistical | HA | 18.78/5.80/16.40 | 21.46/7.48/18.89 | 24.64/9.80/21.86 | 28.46/13.18/25.92 | 22.29/8.28/19.73 |
| | ARIMA | 18.28/5.46/16.03 | 22.33/7.99/19.91 | 26.38/11.19/23.97 | 30.00/14.61/27.76 | 22.93/8.84/20.49 |
| Simple machine learning | SVR | 15.09/3.99/13.41 | 17.21/5.08/15.55 | 19.44/6.32/17.53 | 21.54/7.59/19.50 | 17.54/5.32/15.74 |
| | KNN | 16.29/4.66/13.74 | 18.27/5.86/15.11 | 20.19/7.14/16.72 | 22.49/8.91/18.65 | 18.54/6.18/15.39 |
| Deep learning | LSTM | 14.89/3.96[a]/12.95 | 16.53[a]/4.79[a]/14.75 | 18.22[a]/5.78[a]/15.65 | 19.75[a]/6.74[a]/16.96 | 16.74[a]/4.98[a]/14.50 |
| | GRU | 14.85[a]/3.97/12.64[a] | 16.60/4.90/14.30[a] | 18.41/6.02/15.27[a] | 20.19/7.12/17.01 | 16.88/5.13/14.23 |
| | BILSTM | 14.94/3.97/12.66 | 17.00/5.04/14.67 | 18.84/6.17/15.39 | 20.72/7.36/16.32[a] | 17.21/5.26/14.22[a] |
| | RPConvformer | **14.09/3.47/12.18** | **14.47/3.81/12.42** | **14.91/4.05/12.80** | **15.19/4.19/13.12** | **14.47/3.76/12.44** |

The best results are emphasized in bold font.
[a]The second-best results.

learning is susceptible to interference by local features, and information at other moments is not fully exploited. For KNN, Euclidean distance is used to measure the distance between sequences, ignoring the serial trend correlation. In addition, the prediction may fail if the predicted samples deviate from the training sample space. For SVR, the margin of separation between classes is sensitive to outliers.

Deep learning methods represented by RNN and its variants build a deeper nonlinear network, meanwhile introducing a reasonable gate structure to keep and update the information useful for flow prediction at the next moment, achieving a large improvement in performance. However, conventional deep learning such as LSTM, GRU, and BILSTM, employs recursive processing of sequences, which causes errors to pass and magnify over the course of a recursive cycle, and therefore deteriorates for longer-term predictive performance. Among the above methods, the idea of LSTM is closest to our proposed model. However, there are three important improvements in contrast to LSTM, the first is that our model parallels the sequence and introduces relative position information between nodes in the sequence to improve the prediction

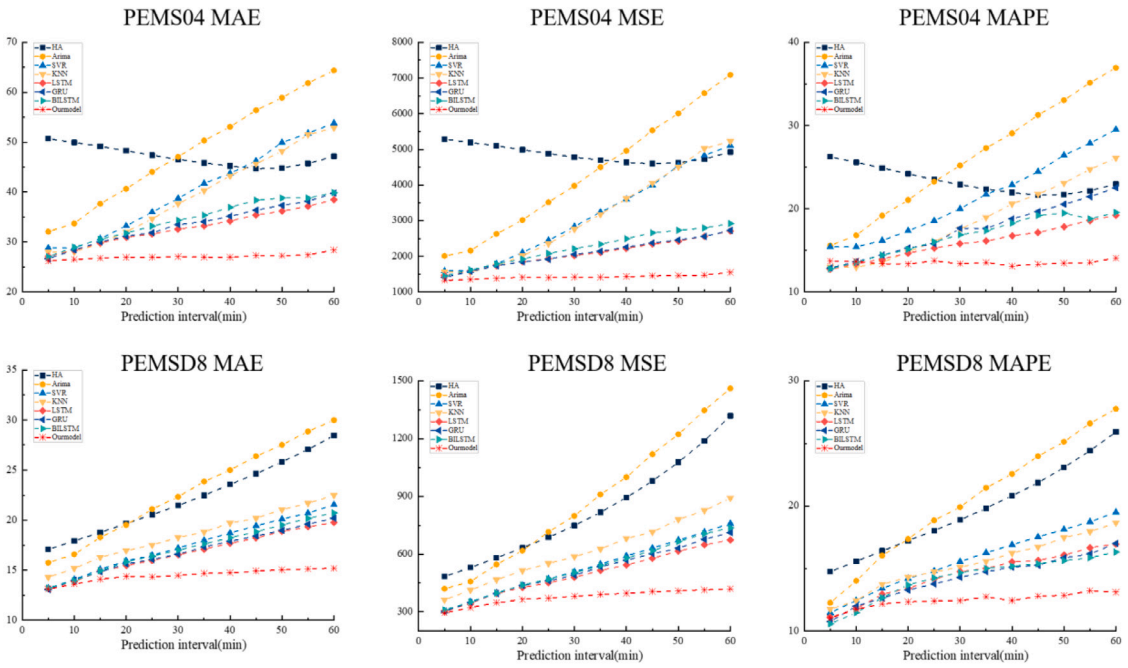


**Fig. 8.** Multi-step prediction comparison of different methods on datasets PEMS04 and PEMSD8.

**Table 5**
The results of ablation study.

| Comparison on datasets of PEMS04(MAE/MSE($\times 10^3$)/MAPE(%)) | | | | | |
|---|---|---|---|---|---|
| Model name | 3 steps | 6 steps | 9 steps | 12 steps | Average error |
| RPConvfomer-noRP | 26.93/1.42/13.52 | 27.34/1.42/13.47 | 27.43/1.48/13.57 | 28.48/1.56/14.15 | 27.19/1.44/13.58 |
| RPConvfomer-no1DCNN | 27.22/1.45/13.40 | 27.55/1.46/13.40 | 27.82/1.50 /13.60 | 28.74/1.56/14.53 | 27.59/1.47/13.70 |
| RPConvformer | **26.78/1.38/13.40** | **27.05/1.42/13.40** | **27.29/1.45/13.35** | **28.42/1.55/14.07** | **27.06/1.42/13.54** |
| Comparison on datasets of PEMSD8(MAE/MSE($\times 10^2$)/MAPE(%)) | | | | | |
| Model name | 3 steps | 6 steps | 9 steps | 12 steps | Average error |
| RPConvfomer-noRP | 14.12/3.52/12.32 | 14.50/3.84/12.53 | 14.94/4.05/12.91 | 15.27/4.23/13.55 | 14.50/3.78/12.71 |
| RPConvfomer-no1DCNN | 15.11/3.98/12.86 | 17.04/5.01/15.02 | 18.61/6.07/15.77 | 20.59/7.29/17.60 | 17.21/5.24/14.72 |
| RPConvformer | **14.09/3.47/12.18** | **14.47/3.81/12.42** | **14.91/4.05/12.80** | **15.19/4.19/13.12** | **14.47/3.76/12.44** |

The best model is shown in bold font.

performance when calculating the attention score matrix. The second is that proposed model is not a complete black box. We can use the attention score matrix to investigate the degree of attention between the sequence nodes. The last is that our model introduces key mask technology so that it can adapt to the input with any length change. Limited historical information is more common in practical engineering applications, and our model is more suitable for engineering problems.

Finally, the testing results of RPConvformer is visualized, as shown in Fig. 9, which indicates that RPConvformer can reasonably fit the trend and seasonal features of fluctuates traffic flow. However, when the traffic suddenly increases or decreases at a certain time node, there is a lag in the model's predictions because it does not have time to learn this change in trend. It is reported in the paper (Liu et al., 2015) that the solution to the above problem is to extract these nodes and then deal with them separately.

In summary, RPConvformer exploits Transformer framework which combines the improvement strategies of convolutional networks and relative positional embedding to fit traffic flow prediction. Experiment results verify the superior of proposed model.

### 4.4. Ablation experiments

It is necessary to investigate the contribution of the improvement modules for Transformer. Specifically, there are two improvements in our model. On the one hand, a pure 1D convolutional framework as a sequence embedding layer to encode the traffic flow sequence. On the other hand, the relative position is introduced in the multi-head attention layer, and the order relationship of time nodes is further considered. Thus, in this section, the performance of two variant models are discussed on PEMS04 and PEMSD8.

- **RPConvformer-noRP:** Relative position is removed from RPConvformer to study the contribution of relative positional embedding. That is, the improved SDPA degenerates into the original SDPA (see Eq. (4)).
- **RPConvformer-no1DCNNs:** the sequence embedding layer block consisting of 1D convolution is removed from RPConvformer to study the contribution of 1D causal convolution.

The super parameters of the above model are consistent with RPConvformer. From Table 5, it can be seen that when removing the relative positional embedding, the average error of the model increased by 0.48%, 1.41%, and 0.30% in terms of MAE, MSE, and MAPE respectively in PEMS04, and when dropping the 1D convolutional network, the average error of the model increased by 1.96%, 3.52%, 1.18% in terms of MAE, MSE, and MAPE respectively in PEMS04, which indicates that relative position and 1D convolution network have little contribution to the model in PEMS04. According to the ablation study on PEMSD8, the 1D convolution network have obvious benefits for improving the performance of Transformer, if it is abandoned, the performance will decrease by 18.94%, 39.36%, and 18.33% in terms of

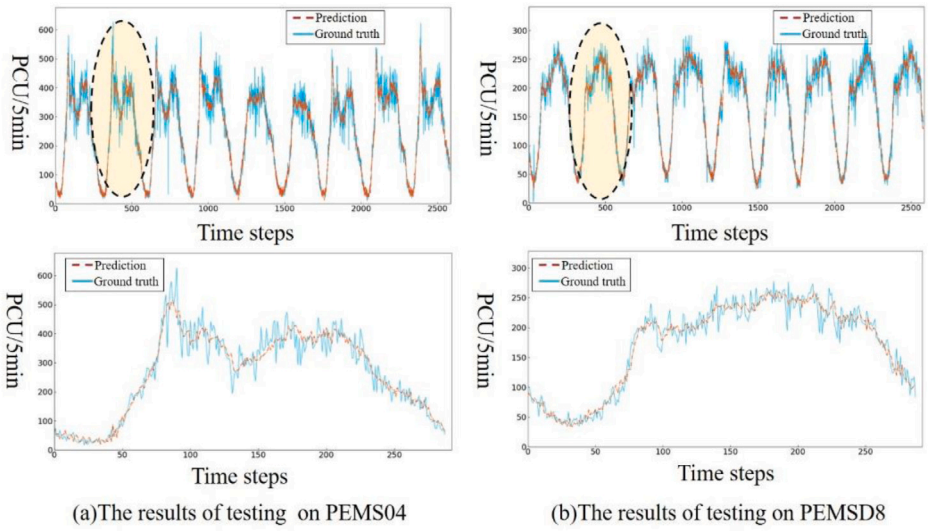


**Fig. 9.** Visualization of RPConvformer prediction testing data results. The red dotted line is the predicted value, and the blue solid is the ground truth, yellow circle transparent box indicates a day of visualization. The first line is the visualization result of 10 days traffic flow prediction, and the second line corresponds to the 1 day traffic flow prediction result of the yellow circle transparent box.

MAE, MSE, and MAPE. On the contrary, relative position has slightly improved the performance of Transformer.

Furthermore, we evaluate the prediction performance of variant models at each time step, and the results are shown in Fig. 10. For the PEMS04, the improved modules have no obvious gain in our model performance. In addition, the performance fluctuates significantly at different time nodes. According to the data analysis in Section 4.1 (see Fig. 6), it can be found that the traffic flow collected by PEMS04 has more frequent fluctuations and more abnormal points, which is the most likely cause of unstable prediction performance. For the PEMSD8, without the convolutional network, the model drops significantly in terms of long-range prediction performance. However, the gain in relative position to the model is not significant.

To sum up, 1D convolutional network as a sequence embedding block can significantly improve the traffic flow prediction performance, while RP does not significantly improve the performance. Therefore, we can pay more attention to how to build a strong CNN-based framework to extract the local context semantic information when it comes to traffic flow prediction.

### 4.5. Random mask analysis

One of the selling points of our model is that it can handle any length of the input sequence because of the KME (see Algorithm 2). Specifically, we set a mask rate $\epsilon$ to simulate the missing traffic flow state in the historical sequence. Assuming that the historical input sequence length is 100, $\epsilon = 0.2$, i.e. 20 time nodes in the input sequence will be masked (the corresponding node is replaced with value of 0). In addition, the position of the mask is random to verify the robustness of the model.

In this section, we set $\epsilon = (0.05, 0.1, 0.15, 0.2)$ to simulate the lack of historical data, and if $\epsilon = 0$ represents the input sequence is complete. The experiment results are shown in Fig. 11. There are two obvious facts: on the one hand, with the increase of the $\epsilon$, the performance of the model decreases in varying degrees, on the other hand, the mask rate does not affect the robustness of the model for long-term prediction. Then, under the random mask experiment, the performance of the model on datasets PEMSD8 is stable to that on datasets PEMS04, this may be because PEMSD8 has more stable traffic flow features than PEMS04 (see Fig. 6), and the limited historical information is enough for the model to learn useful information from the data. Another important conclusion is that when the mask rate $\epsilon$ is greater than 10% (that is, 10% of the historical data is missing), the prediction performance of the model will suffer a Waterloo decline.

In practical engineering applications, the missing traffic flow data is more oriented. Although, some works (Wang, Li, Li, & Yang, 2021; Yuan et al., 2022; Zhang, Zhang et al., 2021) has begun to repair the missing data in traffic flow based on deep learning methods, the repaired data is still noisy. This part of the research shows that when the missing rate is less than 10%, the impact on the prediction performance of the proposed model is slight.

### 4.6. Explanation is all you need

Another selling point of our model is that RPConvformer's multi-head attention mechanism provides a method to understand dependencies between temporal nodes. Specifically, RPConvformer includes 3 types of attention modules. The first attention module is the self-attention module in the encoder (see Algorithm 3), which mainly calculates the attention distribution of the historical sequence. The second attention module is responsible for calculating the attention distribution of the future sequence in the decoder, and the last is the interactive attention module, which is responsible for calculating the attention distribution between the historical and future sequences (see Algorithm 5). We visualized these 3 attention score matrices in the network, as shown in Fig. 12, we can draw the following conclusions::

- **Self-attention in encoder:** The self-attention mechanism in the encoder calculates the attention distribution of the history sequence itself. Partial heads have very similar distribution characteristics (Head1 and Head6), which also indicates that the model has been fully trained. In addition, the recent time node obtains a higher score that indicates that recent historical traffic flow states contribute more to the prediction performance of the model.
- **Self-attention in decoder:** The self-attention mechanism in the decoder focuses on the attention distribution of future sequence itself. The attention score of the nearest time node in the future is higher, because the decoder predicts future traffic flow through autoregressive. In addition, the values of the upper triangle of the matrix are all 0, because the causal mask (CMD, see Algorithm 4) is employed to ensure that future information not be leaked.
- **Interactive-attention in decoder:** The interactive-attention in the decoder performs dependency calculations between historical sequences and future sequences. In fact, the obtained distribution

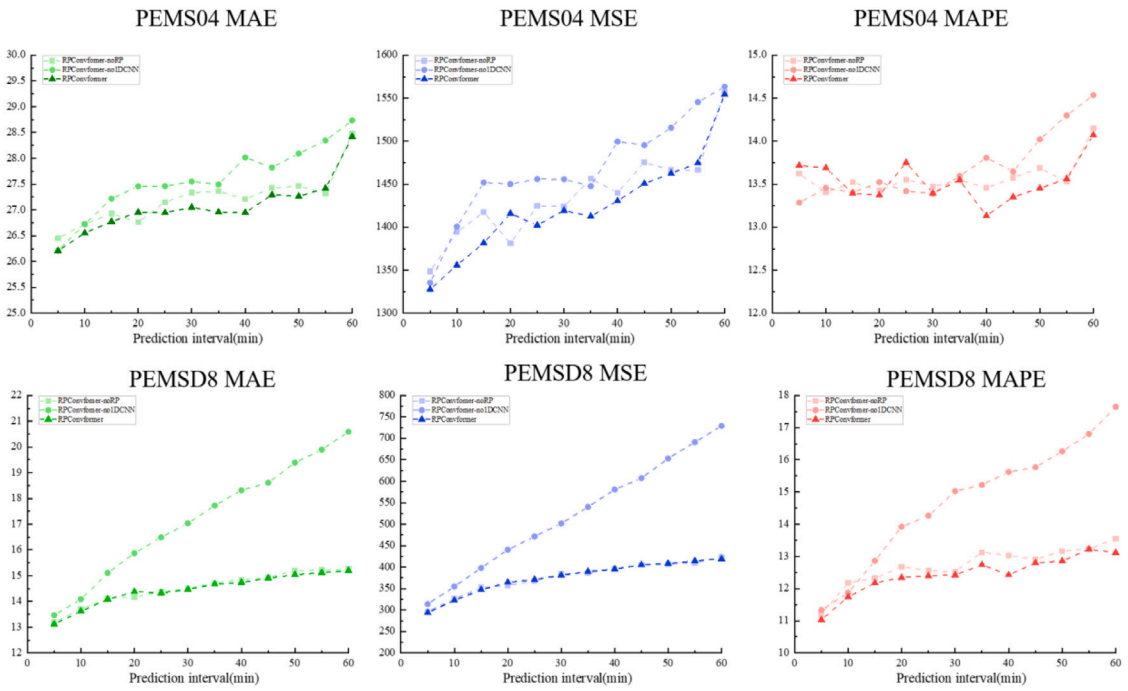


**Fig. 10.** Multi-step prediction comparison of RPConvformer and its variants on PEMS04 and PEMSD8.

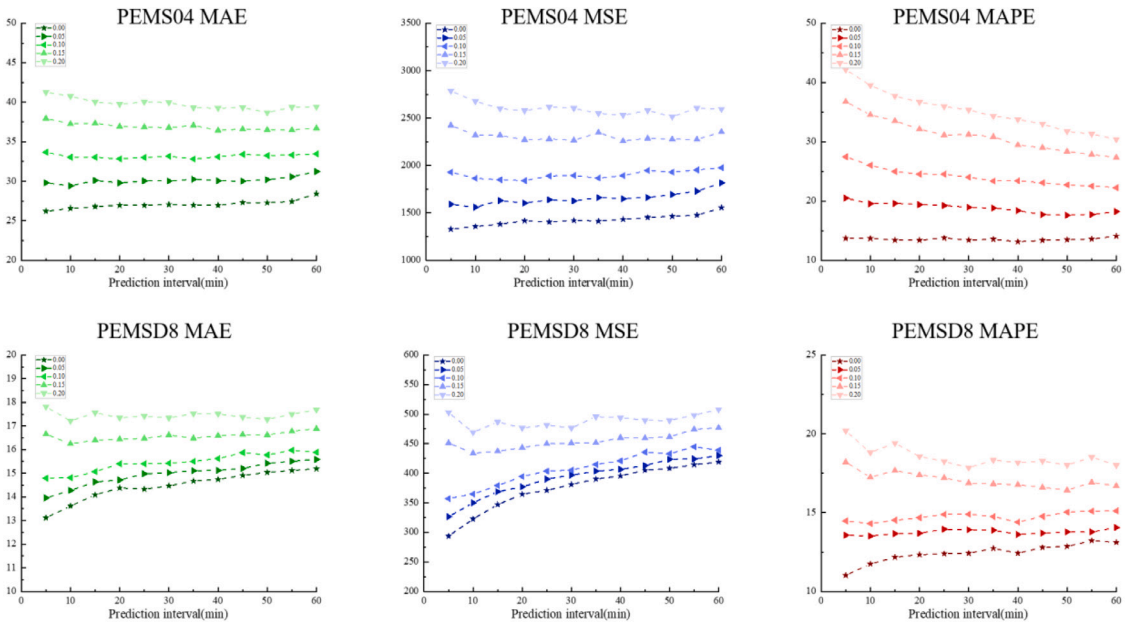


**Fig. 11.** Multi-step prediction comparison of random mask implementation in historical input sequence on PEMS04 and PEMSD8.

matrix is the relationship between $V_F = (v_H, \ldots, v_{H+F-1})$ and $V_H = (v_1, \ldots, v_H)$, due to the right shifted is adopted in teaching forcing to ensure the logic of autoregression. Compared with the self-attention matrix, the distribution of the interaction-attention matrix is richer, which indicates that the relationship between future and historical sequences contributes more to traffic flow prediction. On the one hand, the nearest future traffic state node has a stronger correlation with the historical traffic state, such Head1, Head2, and Head5. On the other hand, the historical traffic state node closest to the future time node has a higher dependence on the future sequence, such as Head3, Head4, and Head8. In summary, the interactive attention module provides two aspects of information, namely the nearest future time node and historical sequence correlation and the terminal historical node and future sequence correlation. The distribution matrix results show that the RPConvformer fully considers the context information of the sequence.

## 5. Conclusion

Traffic flow prediction is one of the important components of intelligent transportation system. In view of the shortcomings of the current traditional deep learning methods in time series modeling, such as the inability to process data in parallel and the lack of interpretability, we adopted a more flexible time series modeling framework called

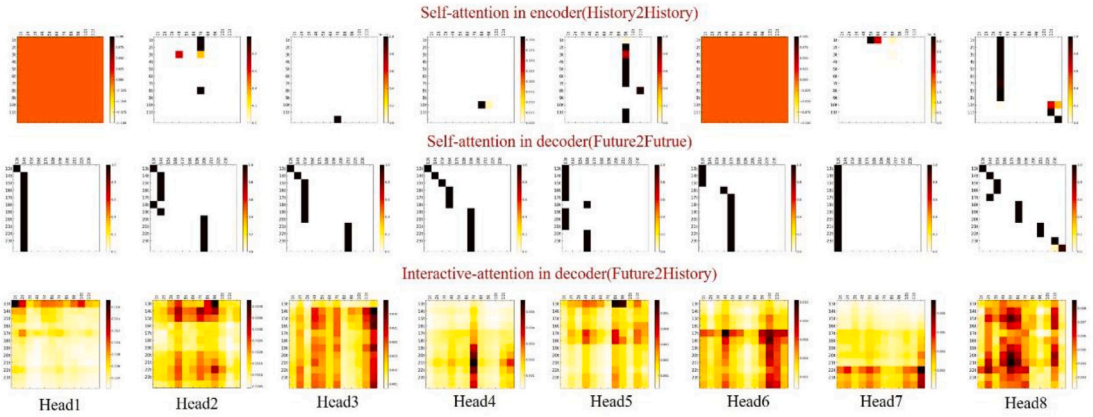


**Fig. 12.** Visualization of attention score matrix. The top row is the self-attention matrix of the historical sequence, the middle row is the self-attention matrix of the future sequence, and the bottom row is the interactive-attention matrix of the future and historical sequence. Each column represents the different distribution in the same layer.

Transformer. Furthermore, we analyzed the shortcomings of the vanilla Transformer, mainly including the failure of position information due to the calculation of attention score, and the lack of local correlation of nodes in the sequence. Thus, we develop a pure 1D CNN network consisting of original 1D CNN and 1D causal CNN as a sequence embedding layer to improve the model's ability to capture local context information, and also introduce relative positional encoding during execution of the attention mechanism to enhance the sequential logic of the sequence. The improved model is called RPConvformer. RPConvformer has achieved significant performance improvement in traffic flow prediction modeling compared with 7 baseline models. The ablation experiment shows that the sequence embedding module built by 1D CNN networks can significantly improve the prediction performance, while the introduction of relative position bias in the calculation of attention score does not significantly improve the prediction performance of the model. Moreover, we also evaluate the performance robustness of RPConvformer in the absence of historical traffic flow state, and the results show that when the historical node missing ratio is less than 10%, the Key mask of the model can effectively alleviate this adverse effect. In addition, the multi-head attention matrix can provide a further explanation for the dependencies between historical nodes and future nodes.

In the future, we will conduct the following further research: in order to adapt to the prediction of spatial–temporal traffic flow in the road network, the spatial feature capture module is introduced on the basis of RPConvformer; In the data processing stage, data augmentation methods, such as random erasing (Zhong, Zheng, Kang, Li, & Yang, 2020), are used to improve the robustness of the model to the prediction of historical data missing in the time series modeling process.

*Statement*

All methods were carried out in accordance with relevant guidelines and regulations. All experimental protocols were approved by a named institutional and/or licensing committee. Informed consent was obtained from all subjects and/or their legal guardian(s).

## CRediT authorship contribution statement

**Yanjie Wen:** Model landing, Formal analysis, Writring – original draft. **Ping Xu:** Writing – review & editing. **Zhihong Li:** Writing – review & editing. **Wangtu Xu:** Writing – review & editing. **Xiaoyu Wang:** Visualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgments

This paper was Supported by National Social Science Fund Project of China (21FGLB014).

## References

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

Bogaerts, T., Masegosa, A. D., Angarita-Zapata, J. S., Onieva, E., & Hellinckx, P. (2020). A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data. *Transportation Research Part C (Emerging Technologies)*, *112*, 62–77.

Castro-Neto, M., Jeong, Y. -S., Jeong, M. -K., & Han, L. D. (2009). Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, *36*(3), 6164–6173.

Chen, C., Petty, K., Skabardonis, A., Varaiya, P., & Jia, Z. (2001). Freeway performance measurement system: Mining loop detector data. *Transportation Research Record*, *1748*(1), 96–102.

Chen, Q., Song, Y., & Zhao, J. (2021). Short-term traffic flow prediction based on improved wavelet neural network. *Neural Computing and Applications*, *33*(14), 8181–8190.

Cheng, S., Lu, F., Peng, P., & Wu, S. (2018). Short-term traffic forecasting: An adaptive ST-KNN model that considers spatial heterogeneity. *Computers, Environment and Urban Systems*, *71*, 186–198.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Cordonnier, J. -B., Loukas, A., & Jaggi, M. (2019). On the relationship between self-attention and convolutional layers. arXiv preprint arXiv:1911.03584.

Cui, Z., Ke, R., Pu, Z., & Wang, Y. (2018). Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:1801.02143.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860.

Devlin, J., Chang, M. -W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Do, L. N. N., Vu, H. L., Vo, B. Q., Liu, Z., & Phung, D. (2019). An effective spatial-temporal attention based neural network for traffic flow prediction. *Transportation Research Part C (Emerging Technologies)*, *108*, 12–28.

Drakulic, D., & Andreoli, J. -M. (2022). Structured time series prediction without structural prior. arXiv preprint arXiv:2202.03539.

Fisk, C. S. (1988). On combining maximum entropy trip matrix estimation with user optimal assignment. *Transportation Research, Part B (Methodological)*, *22*(1), 69–73.

Hamilton, J. D., & Susmel, R. (1994). Autoregressive conditional heteroskedasticity and changes in regime. *Journal of Econometrics*, *64*(1–2), 307–333.

He, P., Liu, X., Gao, J., & Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with disentangled attention. arXiv preprint arXiv:2006.03654.

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

Humayun, M., Almufareh, M. F., & Jhanjhi, N. Z. (2022). Autonomous traffic system for emergency vehicles. *Electronics*, *11*(4), 510.

Humayun, M., Ashfaq, F., Jhanjhi, N. Z., & Alsadun, M. K. (2022). Traffic management: Multi-scale vehicle detection in varying weather conditions using YOLOv4 and spatial pyramid pooling network. *Electronics*, *11*(17), 2748.

Isufi, E., Loukas, A., Simonetto, A., & Leus, G. (2016). Autoregressive moving average graph filtering. *IEEE Transactions on Signal Processing*, *65*(2), 274–288.

Jiang, W. (2022). Internet traffic prediction with deep neural networks. *Internet Technology Letters*, *5*(2), Article e314.

Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, Article 117921.

Jin, K., Wi, J., Lee, E., Kang, S., Kim, S., & Kim, Y. (2021). TrafficBERT: Pre-trained model with large-scale data for long-range traffic flow forecasting. *Expert Systems with Applications*, *186*, Article 115738.

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International conference on machine learning* (pp. 5156–5165). PMLR.

Kaysi, I., Ben-Akiva, M. E., & Koutsopoulos, H. (1993). An integrated approach to vehicle routing and congestion prediction for real-time driver guidance. *Transportation Research Record: Journal of the Transportation Research Board*, *1408*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y. -X., et al. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, *32*.

Liu, X., Yu, H. -F., Dhillon, I., & Hsieh, C. -J. (2020). Learning to encode position for transformer with continuous dynamical model. In *International conference on machine learning* (pp. 6327–6335). PMLR.

Liu, D., Zhao, Y., Xu, H., Sun, Y., Pei, D., Luo, J., et al. (2015). Opprentice: Towards practical and automatic anomaly detection through machine learning. In *Proceedings of the 2015 internet measurement conference* (pp. 211–224).

Liu, Y., Zheng, H., Feng, X., & Chen, Z. (2017). Short-term traffic flow prediction with Conv-LSTM. In *2017 9th International conference on wireless communications and signal processing* (pp. 1–6). IEEE.

Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., & Wang, Y. (2017). Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, *17*(4), 818.

Ma, C., Dai, G., & Zhou, J. (2021). Short-term traffic flow prediction for urban road sections based on time series analysis and LSTM_BILSTM method. *IEEE Transactions on Intelligent Transportation Systems*.

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C (Emerging Technologies)*, *54*, 187–197.

Ma, X., Zhong, H., Li, Y., Ma, J., Cui, Z., & Wang, Y. (2020). Forecasting transportation network speed using deep capsule networks with nested LSTM models. *IEEE Transactions on Intelligent Transportation Systems*, *22*(8), 4813–4824.

Miglani, A., & Kumar, N. (2019). Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges. *Vehicular Communications*, *20*, Article 100184.

Mihaylova, T., & Martins, A. F. T. (2019). Scheduled sampling for transformers. arXiv preprint arXiv:1906.07651.

Qiao, Y., Wang, Y., Ma, C., & Yang, J. (2021). Short-term traffic flow prediction based on 1DCNN-LSTM neural network structure. *Modern Physics Letters B*, *35*(02), Article 2150042.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67.

Reza, S., Ferreira, M. C., Machado, J. J. M., & Tavares, J. M. R. S. (2022). A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks. *Expert Systems with Applications*, *202*, Article 117275.

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. arXiv preprint arXiv:1803.02155.

Shu, W., Cai, K., & Xiong, N. N. (2021). A short-term traffic flow prediction model based on an improved gate recurrent unit neural network. *IEEE Transactions on Intelligent Transportation Systems*.

Stathopoulos, A., & Karlaftis, M. G. (2003). A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C (Emerging Technologies)*, *11*(2), 121–135.

Sun, P., Boukerche, A., & Tao, Y. (2020). SSGRU: A novel hybrid stacked GRU-based traffic volume prediction approach in a road network. *Computer Communications*, *160*, 502–511.

Tang, W., Long, G., Liu, L., Zhou, T., Jiang, J., & Blumenstein, M. (2020). Rethinking 1D-CNN for time series classification: A stronger baseline. arXiv preprint arXiv:2002.10061.

Tian, Y., Zhang, K., Li, J., Lin, X., & Yang, B. (2018). LSTM-based traffic flow prediction with missing data. *Neurocomputing*, *318*, 297–305.

Van Lint, J. W. C., Hoogendoorn, S. P., & van Zuylen, H. J. (2002). Freeway travel time prediction with state-space neural networks: Modeling state-space dynamics with recurrent neural networks. *Transportation Research Record*, *1811*(1), 30–39.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Vijayalakshmi, B., Ramar, K., Jhanjhi, N. Z., Verma, S., Kaliappan, M., Vijayalakshmi, K., et al. (2021). An attention-based deep learning model for traffic flow prediction using spatiotemporal features towards sustainable smart city. *International Journal of Communication Systems*, *34*(3), Article e4609.

Wang, Y., Li, D., Li, X., & Yang, M. (2021). PC-GAIN: Pseudo-label conditional generative adversarial imputation networks for incomplete data. *Neural Networks*, *141*, 395–403.

Williams, B. M. (2001). Multivariate vehicular traffic flow prediction: Evaluation of ARIMAX modeling. *Transportation Research Record*, *1776*(1), 194–200.

Williams, B. M., & Hoel, L. A. (2003). Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of Transportation Engineering*, *129*(6), 664–672.

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

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.

Yan, H., Deng, B., Li, X., & Qiu, X. (2019). TENER: Adapting transformer encoder for named entity recognition. arXiv preprint arXiv:1911.04474.

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

Yuan, Y., Zhang, Y., Wang, B., Peng, Y., Hu, Y., & Yin, B. (2022). STGAN: Spatio-temporal generative adversarial network for traffic data imputation. *IEEE Transactions on Big Data*.

Zhang, Z., Lin, X., Li, M., & Wang, Y. (2021). A customized deep learning approach to integrate network-scale online traffic data imputation and prediction. *Transportation Research Part C (Emerging Technologies)*, *132*, Article 103372.

Zhang, W., Yu, Y., Qi, Y., Shu, F., & Wang, Y. (2019). Short-term traffic flow prediction based on spatio-temporal analysis and CNN deep learning. *Transportmetrica A: Transport Science*, *15*(2), 1688–1711.

Zhang, W., Zhang, P., Yu, Y., Li, X., Biancardo, S. A., & Zhang, J. (2021). Missing data repairs for traffic flow with self-attention generative adversarial imputation net. *IEEE Transactions on Intelligent Transportation Systems*.

Zheng, H., Lin, F., Feng, X., & Chen, Y. (2020). A hybrid deep learning model with attention-based Conv-LSTM networks for short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, *22*(11), 6910–6920.

Zheng, Z., Yang, Y., Liu, J., Dai, H. -N., & Zhang, Y. (2019). Deep and embedded learning approach for traffic flow prediction in urban informatics. *IEEE Transactions on Intelligent Transportation Systems*, *20*(10), 3927–3939.

Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020). Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 34*, (07), (pp. 13001–13008).

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., et al. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 35*, (12), (pp. 11106–11115).