Hindawi
Security and Communication Networks
Volume 2023, Article ID 1363639, 13 pages
https://doi.org/10.1155/2023/1363639

Research Article

Traffic Safety Oriented Multi-Intersection Flow Prediction Based on Transformer and CNN

Tingting Fu,[1] **Qianwen Yu**,[2] **Haksrun Lao**,[3] **Peng Liu**,[1] **and Shaohua Wan**[4]

[1]*School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China*
[2]*HDU-ITMO Joint Institute, Hangzhou Dianzi University, Hangzhou, China*
[3]*Center of Engineering and Design, Chong Cheng Chinese School, Phnom Penh, Cambodia*
[4]*Department of Information Systems, King Abdulaziz University, Jeddah, Saudi Arabia*

Correspondence should be addressed to Haksrun Lao; haksrunlao@hotmail.com

Received 1 July 2022; Revised 22 August 2022; Accepted 31 January 2023; Published 15 February 2023

Academic Editor: Licheng Wang

Intelligent traffic signal control is one of the important means to ensure traffic safety. Effective signal control can make traffic flow fast and smooth, which first needs current and future traffic information. As the control of one intersection may affect adjacent intersections, this paper proposes to predict future traffic flow with consideration of multi-intersections. It can dynamically improve traffic conditions and reduce traffic congestion. Based on various nonlinear spatial relationships at correlated multi-intersections and the potential time-dependent relationship in traffic volume, a traffic flow prediction method named CNNformer which combines transformer with CNN, is proposed. The convolution neural network (CNN) and transformer are used to extract the spatial and temporal features of correlated multiple intersections. The learnable time code is embedded into transformer's location code, and the location information and time information are injected into the model to help it learn the time characteristics of traffic volume. This study also considers the impact of cyclical traffic flow pattern and proposes CNNformer$^+$. In the method, all of the data from the previous time window, as well as the data from the prior week and month, are correspondingly entered into the network. Finally, the output is generated through average pooling, realizing the learning of cyclical traffic flow characteristics. In the experiment, CNNformer$^+$ and the state-of-the-art traffic flow prediction methods are compared using simulated data. The results show that the proposed model outperforms the existing models in prediction accuracy and efficiency.

1. Introduction

Urban traffic is an important factor of urban function layout, which seriously affects the development of the social economy and the improvement of people's living standards. Due to the increase in consumers' purchasing power, there are more and more private cars, and the road density is increasing as well as the traffic safety concerns. Thus, in-depth research on traffic congestion and effective measures to improve traffic efficiency has become a research highlight.

According to the National Urban Car PARC Report, by the first half of 2019, as of June 2019, there were 250 million cars in China, and 66 cities across the country had more than 1 million cars. As the number of cars on the road rises, morning and evening rush hour overcrowding and minor traffic accidents become more and more common. Traffic congestion and accidents are very detrimental to urban development. They will not only increase the time needed for people's travel activities but also adversely affect people's work efficiency and life experience. Moreover, the congestion will lead to increased vehicle exhaust emissions and damage the environment.

In the face of a complex traffic environment, the prediction of traffic flow can improve the utilization rate of urban road resources and reduce the possibility of car accidents. It can also provide accurate traffic guidance information for urban traffic signal control [1], and the design of data dissemination techniques based on the travel of traffic participants [2]. Shortly, 6G will be crucial for communication, resource allocation, and compute offloading [3, 4]. It will also help to collect data for traffic prediction. Through extracting characteristics from the obtained traffic data, traffic prediction methods can improve road safety and intelligent transportation constructions.

In the past decades, many scholars have put forward various traffic flow prediction models and achieved a series of theoretical and applied research results [5]. Most of these research methods are mainly based on statistical models or shallow machine learning methods to describe the evolution of traffic network flow, such as ARIMA [6], ANN [7], and SVR [8]. However, these methods can only be used when the data are relatively stable and linear. The actual traffic flow is extremely variable and will be affected by weather, date, traffic accidents, and other factors. Because the aforementioned elements have an impact on traffic flow, traffic-related time series data typically exhibit nonlinear or rapid change characteristics and are interdependent. In addition, due to the complex traffic network and the increasing number of vehicles, the spatiotemporal sequence traffic data collected based on the Internet of vehicles technology is large in scale and high in latitude, as well as lots of security threatens [9]. Therefore, the traditional methods are difficult to mine the deep relationship between traffic spatiotemporal series data and face a huge bottleneck when being applied in practice. In recent years, deep learning has been proven to be able to effectively extract depth features and has made breakthroughs in image processing, speech recognition, natural language processing, and other fields [10]. Due to the complex nonlinear spatiotemporal correlation between different traffic time series data, the deep learning method is a good choice for traffic flow forecasting tasks.

Because intersections are often interrelated, especially in cities with large traffic flow and short intersection spacing, the congestion of an intersection may affect the traffic distribution and capacity of the whole region. At the same time, the improvement of traffic congestion at a single intersection may aggravate the congestion at adjacent intersections and cannot accurately improve the overall traffic efficiency. Therefore, it is necessary to predict the traffic flow of multiple intersections. There are not many studies on multi-intersection traffic flow prediction, though, and most of the outcomes are not particularly good.

This paper presents a traffic flow prediction method based on transformer and CNN, called CNNformer. In addition, we have made improvements to CNNformer, added the learning of the periodic characteristics of traffic flow, and proposed CNNformer$^+$. The main contributions are as follows:

(i) The traffic flow prediction in this paper is aimed at multiple intersections. Many existing studies are conducted for a single intersection. The improvement of traffic congestion at a single intersection may aggravate the congestion at adjacent intersections, and cannot accurately improve the overall traffic efficiency.

(ii) This work proposes CNNformer, a new multi-intersection traffic flow prediction approach based on transformer and CNN. The flow data of correlated multiple intersections are constructed into a two-dimensional matrix with the shape of (number of intersections × number of lanes), and CNN is used to extract the spatial features of correlated multiple intersections. The transformer model is innovatively used in intersection flow prediction. Compared with LSTM, transformer can avoid recursion, which allows parallel computing, reduces training time, reduces performance degradation due to long-term dependence, and has better performance in prediction accuracy. In this paper, the learnable time code is embedded into the transformer's location code, and the location information and time information are injected into the model to help the model better learn the time characteristics of traffic volume.

(iii) In addition, this study also considers the weekly and monthly cycle trend of traffic volume, makes improvements on CNNformer, and proposes CNNformer$^+$. CNNformer$^+$ can learn the periodic characteristics of traffic flow.

2. Related Work

With the development of the intelligent transportation systems, many cameras, sensors, and other information collection equipments are deployed on the road. These equipments have accumulated a large number of traffic time series data with spatial information such as traffic flow, vehicle speed, and lane occupancy rate, providing a good data foundation for traffic flow prediction.

2.1. Shallow Machine Learning Methods. For a long time, to improve the congestion analysis and management decision-making ability of intelligent transportation, researchers have proposed a large number of traffic flow prediction models. Willams and Hoel used ARIMA [6] to model the traffic flow. This method is to model the single variable traffic flow sequence data as an autoregressive moving average process, to predict the traffic flow. Chan et al. proposed an ANN model using a mixed exponential smoothing strategy and Levenberg–Marquardt optimization to support short-term traffic flow prediction [7]. Alkheder et al. proposed to use a Bayesian joint neural network for short-term traffic flow prediction of adjacent intersections [11]. Alajali et al. proposed to use gradient enhanced regression tree (GBRT) and random forest (RF) to realize traffic flow prediction and suggested to use extreme gradient enhanced tree (XGBoost) algorithm to process traffic flow big data [12]. However, these methods have not achieved a completely satisfying result.

2.2. Deep Learning Methods. Due to the complex traffic network and the increasing number of vehicles, the usually collected spatiotemporal sequence data related to traffic flow has the characteristics of large-scale, high-dimensional, dynamic, and abrupt. Traditional methods are facing great challenges. The capacity of deep learning to capture nonlinear depth characteristics has raised significant concerns [13]. It can also be used to automatically extract and learn deep-seated features in traffic time series data. Therefore, more and more scholars began to study the traffic flow prediction model based on deep learning [14]. For example, Tu et al. proposed a traffic congestion prediction model SG-CNN [15]. By analyzing the characteristics of traffic data, the model groups road segments. According to the correlation characteristics of road segments in the road network, the CNN model is used to extract the characteristics of road segment data, to realize the information sharing between road segments. Ren et al. proposed a new global-local time convolution network (GL-TCN) to predict traffic flow [16]. This new local time convolution mechanism can effectively capture the local characteristics of long-term traffic flow. At the same time, the influence of the periodicity of the global traffic flow on the local traffic flow law is considered.

Ma et al. proposed a traffic flow prediction model based on a bidirectional LSTM network. By improving the LSTM model, combining the characteristics of sequence data and the long-term dependence of BiLSTM, the bidirectional long-term memory network (BiLSTM) is integrated into the prediction model [17]. Du et al. proposed a deep irregular convolution residual LSTM network model (DST-ICRL) for traffic flow prediction [18]. To learn the spatiotemporal feature representation, the traffic flow between various roads in the road network is modeled as a multichannel matrix, which is comparable to the RGB pixel matrix of the image. Furthermore, deep learning methods, such as Deep Q-learning Network (DQN), can also be used to find optimal offloading strategies in intelligent-connected vehicles [19].

The intersection is the most complex part of the road network because it involves a variety of different objects, such as vehicles and pedestrians. With the increase in traffic demand, the problem of traffic congestion at urban intersections is becoming more and more serious. The short-term traffic flow forecast of intersections has also been the subject of numerous corresponding studies. For example, Qu et al. established a two-layer superposition model based on intersection short-term traffic flow prediction by integrating k-nearest neighbor (KNN) and Elman neural network modeling methods [20]. Kim and Jeong proposed a collaborative traffic signal control method based on multi-intersection traffic flow prediction (TFP-CTSC) [21]. Li et al. proposed a new deep intersection spatiotemporal network (DISTN) for traffic flow prediction. Considering the spatial and temporal characteristics of the convolutional neural network (CNN) and long-term and short-term memory (LSTM), the depth learning method was applied to intersection traffic volume prediction [22]. Furthermore, digital twins have been used to facilitate the design, evaluation, and deployment of IoV-based systems [23, 24]. However, the research is still in an initial stage.

3. Methodology

In terms of traffic flow prediction, the camera on the road is usually used to count the number of cars passing by. If multiple intersections in a certain area are considered, data from different cameras will contain geolocation and time information. Therefore, we can regard the traffic flow prediction problem as a spatiotemporal sequence problem, namely, we can use the time and space information contained in the data to predict the traffic flow of different intersections. The structure of the model is shown in Figure 1. The input of CNNformer $^{+}$ contains the traffic flow data of three time windows, which are the traffic flow data of the previous time window $(X_{t-H}, X_{t-H+1}, \ldots, X_t)$, the simultaneous data of the week before the previous time window $(X_{t-H-\text{week}}, X_{t-H-\text{week}+1}, \ldots, X_{t-\text{week}})$, and the simultaneous data of the previous month in the previous time window $(X_{t-H-\text{month}}, X_{t-H-\text{month}+1}, \ldots, X_{t-\text{month}})$. Each time window contains H time steps, and the traffic flow data of each time step can be described as a two-dimensional matrix. The three input time windows are processed separately, that is, to stack the H two-dimensional matrices in the data of each time window and input them to CNN. After using CNN to extract the spatial features of data, the convoluted data are input into transformer. After using transformer to extract the time characteristics of data, the data of all time steps will be output. Then, it stacks the outputs $(Z_{\text{now}}, Z_{\text{week}}, Z_{\text{month}})$ of the three time windows and puts them into the average pooling layer. The final output of the model is the predicted traffic flow in the next time window.

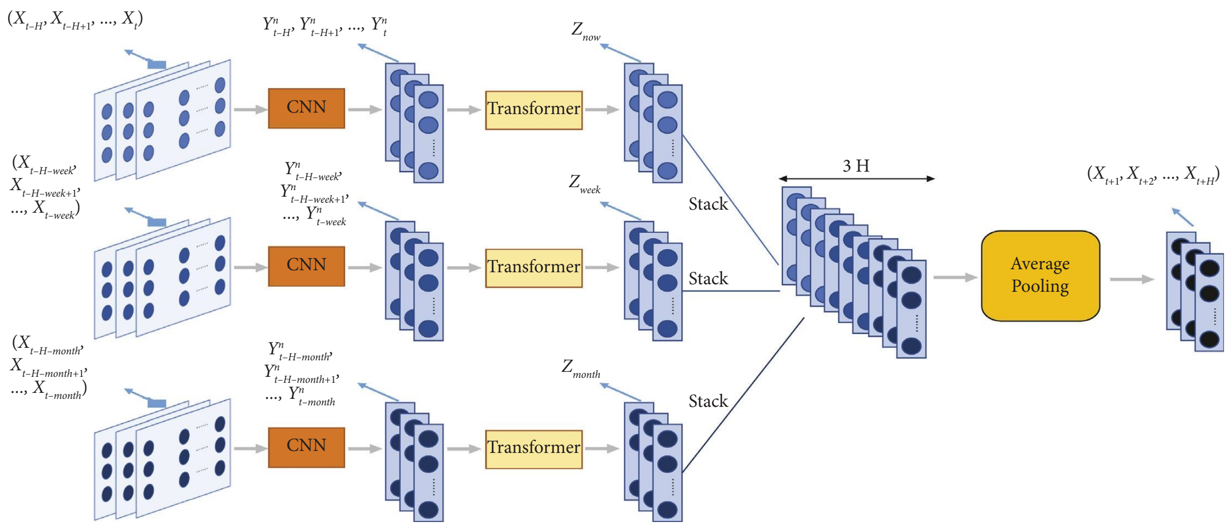
3.1. Extracting Spatial Features Using CNN. Because the intersections are often interrelated, the upstream and downstream intersections may affect the traffic flow prediction of the target intersection. In this paper, CNN is used to extract the spatial features of associated intersections. The input data has the following shape: $[H, N, D]$, where H stands for the number of time steps, N for the number of intersections, and D for the quantity of traffic flow directions at each intersection, where D is equal to 12.

With the great success of convolutional neural network in the field of image processing, other fields are also trying to use the method of deep learning to solve practical application problems. In the field of traffic flow prediction, because the traffic flow based on region or station can be organized into a two-dimensional vector or a one-dimensional vector, it is considered as an effective method to mine the spatial characteristics of traffic volume data using the convolution neural network.

For instance, in time step t, the historical flow data of a given road network can be described as a matrix as follows:

$$X_t = \begin{bmatrix} X_t^{1-1} & X_t^{1-2} & \cdots & X_t^{1-12} \\ X_t^{2-1} & X_t^{2-2} & \cdots & X_t^{2-12} \\ \vdots & \vdots & \ddots & \vdots \\ X_t^{N-1} & X_t^{N-2} & \cdots & X_t^{N-12} \end{bmatrix}. \quad (1)$$

For each element in the matrix, the superscript format is (intersection number - traffic flow direction number), and the subscript represents the time step t. Each row of the matrix represents the traffic flow of all traffic flow directions


$(X_{t-H}, X_{t-H+1}, \ldots, X_t)$
$(X_{t-H-week}, X_{t-H-week+1}, \ldots, X_{t-week})$
$(X_{t-H-month}, X_{t-H-month+1}, \ldots, X_{t-month})$
CNN
$Y^n_{t-H}, Y^n_{t-H+1}, \ldots, Y^n_t$
$Y^n_{t-H-week}, Y^n_{t-H-week+1}, \ldots, Y^n_{t-week}$
$Y^n_{t-H-month}, Y^n_{t-H-month+1}, \ldots, Y^n_{t-month}$
Transformer
Z_{now}
Z_{week}
Z_{month}
Stack
3 H
Average Pooling
$(X_{t+1}, X_{t+2}, \ldots, X_{t+H})$
FIGURE 1: The CNNformer$^+$ model structure.

at time t at the nth target intersection. Each intersection has 12 traffic flow directions. Therefore, there are 12 columns of traffic flow data, and each column of the matrix represents the traffic volume in a certain traffic flow direction from intersection 1 to intersection N.

When the data of a time step can be described as a matrix, it is easy to think that the matrix can be used as the input of CNN. The convolution model in this paper is shown in Figure 2. The spatial features of associated intersections are extracted by using two-dimensional convolution layers with a convolution kernel size of (2, 2) and padding size of (2, 1). After convolution, a ReLU and Dropout layer are added.

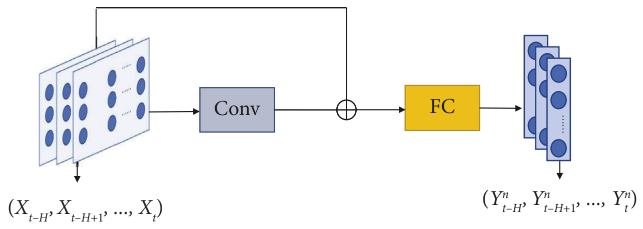
The output X of the Nth convolution layer at time t is X_t^N, it will then pass through a residual connection. Finally, through the full connection layer, it is transformed into a one-dimensional spatial eigenvector Y_t. This vector is used as the input of the transformer network to capture the time correlation.

The output shape of CNN is $[H, M]$, where M represents the sum of traffic flow directions at each time step and all relevant intersections, and H represents the number of time steps.

3.2. *Extracting Time Characteristics Using Transformer.*

The task of predicting traffic flow is a typical time series prediction challenge that uses historical observation data to forecast future traffic flow data. Since transformer is an excellent sequence model, this paper takes the output of CNN as the input of transformer and uses transformer to extract the time characteristics of traffic flow.

Currently, the majority of tasks involving traffic flow forecasting uses RNN and its derivatives, LSTM and GRU. RNN and its variants must process data in sequence during training. The calculation of time step t depends on the calculation result at time $t-1$, so parallel training is not possible. In addition, the coding of the traffic flow by RNN and its variants is only retained in the next time step, which


Conv
FC
$(X_{t-H}, X_{t-H+1}, \ldots, X_t)$
$(Y^n_{t-H}, Y^n_{t-H+1}, \ldots, Y^n_t)$
FIGURE 2: Convolution model structure.

means that the coding of the current time step only strongly affects the representation of the next time step, and its influence will disappear soon after a few time steps. Although the structure of gate mechanisms such as LSTM alleviates the problem of long-term dependence to some extent, LSTM is still powerless for particularly long dependencies. The transformer model can avoid recursion, allows parallel computing to reduce training time, and reduces performance degradation caused by long-term dependency. Compared with RNN and variants, the transformer model has stronger structural flexibility and versatility, and can capture a wider range of information relevance. In addition, in the NLP field, the transformer model processes sentences in a nonsequential manner, and sentences are processed as a whole rather than word by word.

The transformer does not rely on past hidden states to capture dependencies on previous words but processes a sentence as a whole, so there is no risk of losing or forgetting past information. Based on the abovementioned advantages, this paper attempts to apply transformer to the task of traffic flow prediction.

The input of transformer is a sequence of spatial eigenvectors containing H time steps, expressed as $(Y_{t-H}, Y_{t-H+1}, \ldots, Y_t)$, where Y_t is the spatial eigenvector output from the flow data of time step t after n convolution layers, where $t-H$ to t is the historical time step. The network is trained to predict the traffic flow of all associated intersections in the next H time steps.

Transformer is a seq2seq model. The encoder layer receives input and the decoder layer obtains output.

3.2.1. Encoder. The encoder layer of transformer includes two sublayers:

(i) The first sublayer is a multihead attention, which is used to calculate the input Self-Attention.

(ii) The second sublayer is feed forward, which is a simple fully connected network.

After each sublayer, the residual network is simulated, and the results of each sublayer are displayed as follows:

$$\text{LayerNorm}(x + \text{Sublayer}(x)), \tag{2}$$

where Sublayer (x) represents the mapping of the sublayer to the input X. To ensure full connection, the dimensions of the output of all sublayers and embedded layers are the same.

The structure of the encoder layer is shown in Figure 3. The encoder input consists of the following three parts:

(i) Input embedding: In the original transformer model, the input of the model is a high-dimensional eigenvector. The feature vector is obtained by converting the input text through word embedding method such as Word2Vec [25], which is called an embedded vector. This paper uses the full join layer to replace the word embedding method to encode the input data. After the full join layer, the shape of the input data becomes $[H, E]$, where H represents the number of input time steps and E represents the feature size of the input data.

(ii) Position encoding: Transformer adds an additional vector positional encoding to the input of the encoder layer. The dimension of this vector is the same as that of the embedded vector, which is used to provide relative position information. This vector can determine the position of the current time step in the time window, and the transformer can learn the position information of the time step through this vector. The formula of the position code is shown as follows:

$$PE(pos, 2) = \sin\left(pos/10000^{2i}/d_{\text{model}}\right), \tag{3}$$

$$PE(pos, 2i + 1) = \cos\left(pos/10000^{2i}/d_{\text{model}}\right), \tag{4}$$

where pos refers to the position of the current time step in the time window, i refers to the subscript of each value in the vector, and d_{model} refers to the size of the input dimension. When pos is an even number, Sine coding is used; when pos is an odd number, Cosine coding is used.

(iii) Global time encoding: Based on the transformer model, this paper not only uses location coding for local location embedding but also takes into account the effectiveness of timestamp information in practical applications. The location codes are extracted from the timestamp corresponding to time series data.

The calculation of global time encoding is shown as follows:

$$GTE = FC(X_{\text{mon}}, X_{\text{dow}}, X_d, X_h, X_{\text{min}}), \tag{5}$$

where X_{mon} refers to the month location embedding, X_{dow} refers to the day of week location embedding, X_d refers to the day location embedding, X_h refers to the hour location embedding, and X_{min} refers to the minute location embedding. These five vectors are combined and input into the full connection layer for coding to generate a learnable embedding.

Finally, the model adds the abovementioned three embedded vectors and sends them to the next layer as input.

A multihead attention is equivalent to the integration of M Self-Attention. The specific process of Self-Attention is as follows:

(i) Self-Attention will use the input embedded vector to calculate three new vectors. The dimension of the vector is the same as that of the embedded vector. These three vectors are named as Query, Key, and Value, respectively. These three vectors are obtained by multiplying the embedded vector with a matrix, which is randomly initialized. The dimension of the matrix is $[64, E]$, and E represents the characteristic size of the input data.

(ii) Calculate the score of Self-Attention, which determines the degree of attention paid to the input data of other time steps when the model encodes one-time step data at a certain position. The fractional value is calculated by point multiplication of Query and Key.

(iii) Next, divide the result of point multiplication by a constant. The constant value selected in this paper is 8, which is the root of the first dimension of the matrix. Then, do a Softmax calculation on the obtained results. The result is the correlation between each time step data and the time step data at the current location.

(iv) Finally, use the result to multiply the value to get the Self-Attention Value.

This method of determining the weight distribution of values through the similarity between Query and Key is called scaled dot product attention. The calculation formula is shown as follows:

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{6}$$

where d_k represents the dimensions of Query, Key, and Value vectors.

A multihead attention is to perform the process of scaled dot-product attention M times, in which not only one group of Q, K, and V matrices is initialized, but M groups are initialized, and then M matrices are output.

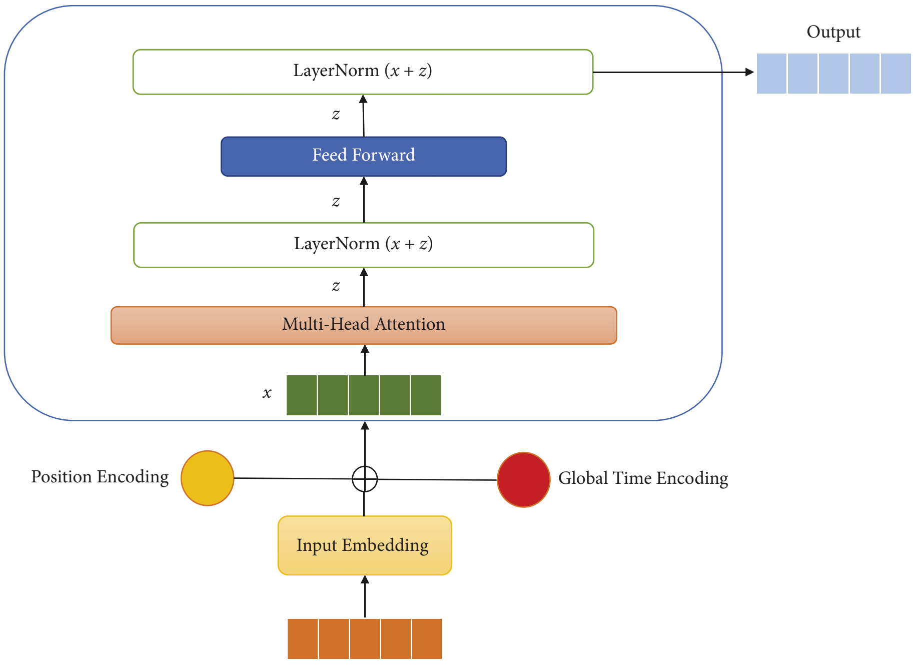


FIGURE 3: The transformer encoder layer.

However, the feed forward neural network cannot input multiple matrices. Therefore, M matrices need to be reduced to one. The precise method entails joining M matrices to create a large matrix, multiplying this large matrix by a weight matrix with the random initialization, and then obtaining the final matrix.

In the transformer, each sublayer will be followed by an incomplete module, and there is a layer normalization. There are many normalization methods, but the purpose of each method is to normalize the input data to achieve the effect that the mean value is 0 and the variance is 1. The data should be normalized before entering the activation function so that the input data do not fall in the saturation region of the activation function.

Unlike batch normalization, which calculates the mean and variance in the batch direction, layer normalization calculates the mean and variance on each sample. Therefore, layer normalization is usually used to normalize the sequence model.

3.2.2. Decoder.

The transformer decoder layer includes three sublayers.

(i) The first sublayer is masked multihead attention, which is also the Self-Attention of calculation input. However, since future information cannot be known at the time of generation, it is necessary to mask future information. For a sequence, suppose the time step is t, the decoding output should only depend on the output before t, not after t. Therefore, mask operation is required.

(ii) The second sublayer is encoder-decoder attention. The output of the encoder layer and the output of the masked multihead attention sublayer are used for attention calculation.

(iii) The third sublayer is feed forward, which is the same as the encoder layer.

The structure of the decoder layer is shown in Figure 4. The traffic flow of the input decoder layer is composed of a part of the historical data that is close to the predicted data and an empty vector. The length of the empty vector is the length of the data to be predicted. The encoder layer uses the same coding technique for input traffic volume.

The masked multihead attention sublayer of the decoder layer needs to use a mask so that the decoder cannot see future information. The specific method is to generate an upper triangular matrix, the values of which are all 0 s, and apply this matrix to each sequence to achieve the purpose of covering.

The encoder-decoder attention sublayer of the decoder layer uses the output information of the encoder to calculate the content of the current decoded output. The difference between this part and Self-Attention lies in the three vectors of Q, K, and V. Q is the attribute of the decoder, while K and V are the last output K and V of the encoder layer. The calculation method of attention is the same as that of Self-Attention. Through this method, the encoder can capture the output information of the encoder.

3.3. *Learn the Periodicity of Traffic Flow Using Average Pooling.*

When the decoder layer is completely executed, the final output of the three time windows is Z_{now}, Z_0, and Z_{month}. Then, we stack these three vectors and input them to the average pooling layer. The calculation process of average pooling is as follows:

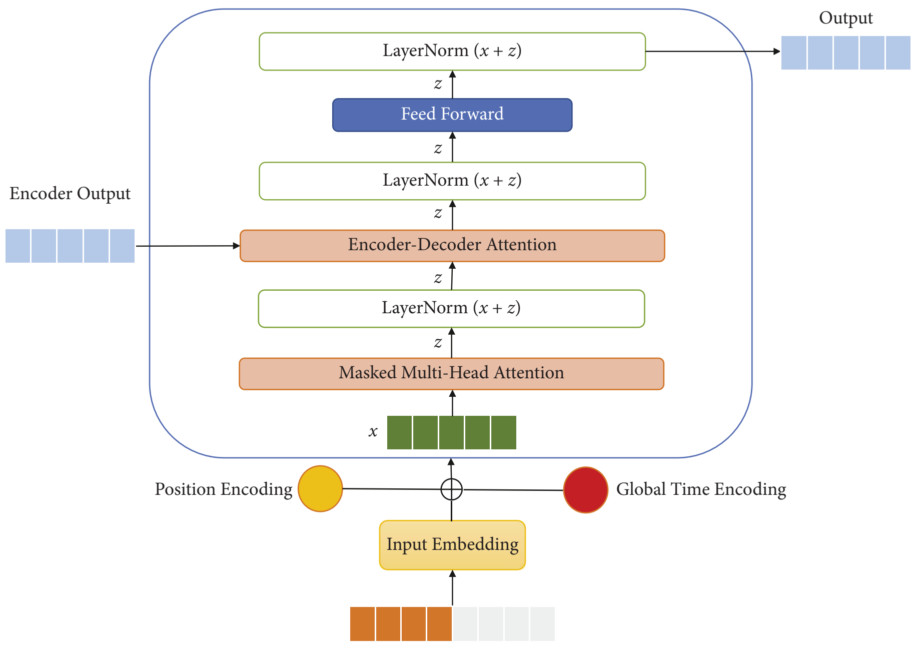


FIGURE 4: The transformer decoder layer.

$$\hat{q}_{t+1} = \mathrm{AvgPooling}\left(Z_{\mathrm{now}}, Z_{\mathrm{week}}, Z_{\mathrm{month}}\right), \tag{7}$$

where $\hat{q}_{t+1}$ represents the predicted flow data. As shown in Figure 5, average pooling involves combining feature points from different neighborhoods and averaging their values to create new features. Compared with the full connection layer, the average pooling can greatly reduce the network parameters, thus reducing the overfitting phenomenon.

The final output of the average pooling layer is the predicted traffic volume of the next time window. Gaussian error linear element (GELU) is used as the activation function of the average pooling layer. It is a high-performance neural network activation function because the nonlinear change of GELU is a random regular transformation mode that meets the expectation, and the formula is as follows:

$$xP\left(X \le x\right) = x\Phi\left(x\right), \tag{8}$$

where $\Phi(x)$ refers to the cumulative distribution of the Gaussian normal distribution of x. GELU introduces the idea of random regularity in activation, which is a probabilistic description of neuron input, and is more intuitive and natural.

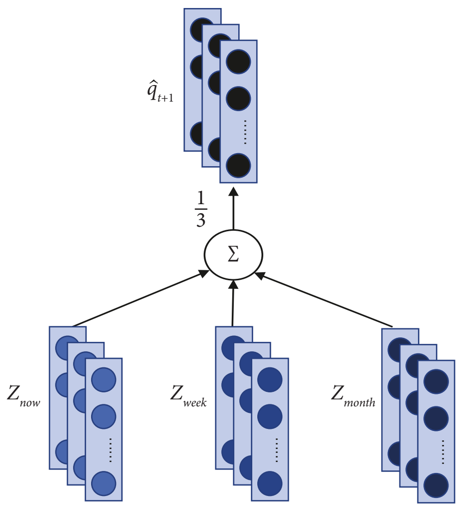


FIGURE 5: Architecture of average pooling.

3.4. Loss Function. The loss function, also known as the error function, is used to measure the operation of the algorithm. The loss function is shown as follows:

$$L\left(\alpha\right) = Loss\left(\hat{q}_{t+1} - q_{t+1}\right), \tag{9}$$

where α represents the learning rate, Loss() represents the loss function, $\hat{q}_{t+1}$ represents the predicted flow data, and q_{t+1} represents the actual flow data. The error between the anticipated traffic flow in the following time window and the actual traffic flow in that time window is measured using the loss function to determine how closely the predicted output value is to the actual value.

3.5. Optimization Algorithm. The application of machine learning is a process highly dependent on experience. With a large number of iterations, many models need to be trained to find the right one. When training a neural network, we frequently employ a large data collection, which will cause the training time to be extremely slow. Therefore, using an appropriate optimization algorithm can effectively improve the speed of the training model. Gradient descent is a method to find the objective function, that is, to minimize the loss function. It uses gradient information to find the appropriate objective value by iteratively adjusting parameters. It is one of the most widely used optimization algorithms in neural networks. This paper uses Adam as the optimization algorithm of the model. The reason is that it is essentially the combination of momentum and RMSprop algorithms and then corrects its deviation. The momentum algorithm uses momentum similar to physics to accumulate gradients, and the RMSprop algorithm can make convergence faster while making fluctuations smaller. Therefore, the performance achieved by combining these two algorithms is assumed to be better. Adam fully utilizes the second moment mean of the gradient in addition to computing the adaptive parameter learning rate based on the first moment mean, as does the RMSProp algorithm. Specifically, the algorithm computes exponential moving averages of the gradients, using hyperparameters beta1 and beta2 to control the decay rate of these moving averages. Because the initial moving average, beta1 and beta2 values are all close to 1, the moment estimate's deviation is close to 0. By first computing the deviated estimate, and then, the deviate-corrected estimate, the deviation is optimized.

4. Simulation Experiment of Regional Traffic Flow Prediction Based on AnyLogic

AnyLogic is a professional virtual prototyping environment for designing complex systems with discrete, continuous, and mixed behaviors. Using AnyLogic, one may easily create a simulation model of the intended system and the system's surrounding environment, including its physical equipment and operators. The road traffic Library in AnyLogic allows users to model, simulate, and visualize vehicle traffic. The library supports detailed and efficient physical hierarchical modeling of vehicle motion. AnyLogic can be applied to model vehicles, roads, and lanes of highway traffic, street traffic, production site transportation, parking lot, or any other system.

4.1. Data Description. In the experiment, AnyLogic is used to build a regional road network micro model to simulate the actual road conditions for the statistics of intersection traffic flow data. This area is a real region composed of three associated intersections, and each intersection has 12 lanes, as shown in Figure 6. The simulation data includes three months' traffic flow data. The statistical interval is 15 minutes, and the traffic flow data of all intersections are collected every 15 minutes. Each model data represents the number of vehicles passing in the direction of traffic flow within 15 minutes.

In the simulation, external factors such as morning peak, weekends, and holidays, are considered to enhance the randomness, making the simulation data tend to the real data.

4.2. Data Preprocessing. Before inputting the data into the model, it is necessary to standardize the data to scale the attributes of a sample to a specified range. It is necessary to eliminate the influence of different attributes of samples with different orders of magnitude because

(i) The difference in orders of magnitude will lead to the dominant position of attributes with larger orders of magnitude;

(ii) The difference of orders of magnitude will cause the convergence speed of iteration to slow down;

(iii) Algorithms that depend on sample distance are very sensitive to the order of magnitude of data.

In this paper, min-max standardization, also known as normalization, is used as the method of data standardization. The specific method is as follows: after the data (x) are centered according to the minimum value, it is scaled according to the range (maximum value-minimum value), and the data are converged to [0, 1]. After normalization, the range of the optimization process becomes smaller, the optimization process becomes gentle, and it is easier to correctly converge to the optimal solution. The calculation formula is shown as follows:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \tag{10}$$

4.3. Evaluation Metrics. This paper measures the prediction effect of the model using the mean square error (MSE) and mean absolute error (MAE) loss functions to evaluate the prediction performance of the algorithm more thoroughly.

4.4. Experimental Setup. This paper uses the Python3.7 simulation environment and the deep learning framework PyTorch to build the model. The CPU model used is Intel (R) Xeon (R) w-2133 CPU @ 3.60 GHz, the memory is 32 GB, the GPU model is NVIDIA GeForce GTX 1080 Ti, and the operating system is Ubuntu.

4.5. Simulation Results and Analysis. The proposed CNNformer$^+$ is compared with several baseline models, including CNN, LSTM, DISTN [22], CNNformer, transformer, and informer [26]. Table 1 compares the performance of the baseline model and CNNformer$^+$ in the traffic flow prediction task at the associated intersections. Table 2 shows the hyperparameter settings of the experimental model.

The following phenomena were observed during the experiment:

(i) Compared with the traditional time series model LSTM, the convolution model combined with CNN and LSTM is more suitable for traffic flow prediction tasks. This is because CNN can better extract the spatial features of associated intersections than LSTM. Compared with CNN, LSTM has better performance in traffic flow prediction tasks.

(ii) Compared with the model combining CNN and LSTM (DISTN). Transformer is more suitable for traffic flow prediction tasks, thanks to its ability to better establish long-distance dependencies, and

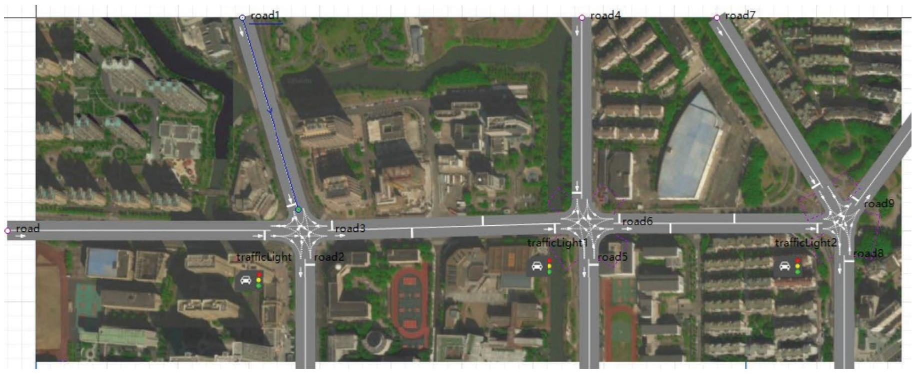


Figure 6: The road network in the simulation network contains three intersections (Yinzhou district, Ningbo city, China).

Table 1: Comparison of simulation results.

Models	MAE	MSE
CNN	0.12205	0.02899
LSTM	0.09840	0.01720
DISTN	0.09778	0.01683
Transformer	0.09599	0.01656
Informer	0.09736	0.01681
CNNformer	0.09499	0.01604
CNNformer$^+$	0.09220	0.01515

Table 2: Experimental model hyperparameter settings.

Parameters	Parameters size
Input dimensions	512
Batch size	128
Learning rate	0.0001
Epoch	120
Time window size	96
Number of encoder layers	1
Number of decoder layers	1

unlike LSTM, which depends on the calculation at the previous moment, it can be well parallel.

(iii) Informer has achieved good results in time series prediction tasks in many fields, which proves that the improvement made by informer in transformer is effective. However, in this experiment, the accuracy of the forecast is lower than that of transformer, which might be because informer has a difficult time capturing the details of traffic flow data.

(iv) Compared with transformer, CNNformer has higher prediction accuracy, thanks to CNN's ability to extract the spatial features of traffic flow data at associated intersections.

(v) The prediction accuracy of CNNformer$^+$ is higher than that of CNNformer, which verifies that learning the periodic characteristics of traffic flow is helpful to improve the prediction accuracy.

(vi) The model proposed in this paper achieves the best results in the traffic flow prediction task, which shows that the model is superior to some of the most advanced traffic flow prediction methods in the literature.

It can be seen from Figure 7 that the dimension size of the hidden layer inside the model will also affect the performance to a certain extent. The richer hidden layers can play a positive role. However, when the number of hidden layer units is greater than 512, the model performance begins to decline.

Figure 8 shows the comparison between the real traffic volume and the traffic volume predicted by CNNformer$^+$ at a single time step, i.e., 10 a.m., 2 p.m., and 5 p.m. Each time step includes $36 (12 \times 3)$ traffic movements. As can be observed, the model successfully captures the changing trend of the actual traffic volume in the majority of traffic flow directions where the predicted value is near the real value.

Figure 9 shows the comparison of real traffic volume at 10 a.m. with traffic volume predicted by informer and transformer. It can be seen from the marks in the figure that CNNformer$^+$, informer, and transformer have a huge deviation when predicting the traffic volume with movement number 3. However, when predicting the traffic volume with movement numbers 23–28, CNNformer$^+$ can better fit the real traffic volume than informer and transformer, which reflects the superiority of the algorithm used in this model.

After introducing the overall performance of the proposed model, the prediction accuracy of single vehicle flow motion is now given. Table 3 provides the prediction accuracy for each movement at the second intersection. The MSE of the traffic movement from east and west is better than that from north and south. This is because intersection 2 is located in the middle of the three intersections. Since the volume of traffic leaving from the north and south is lower than that leaving from the east to west, the trends of the traffic flow are more varied, which makes it more difficult to predict the direction of the traffic flow.

Table 4 provides the prediction accuracy of each of the three intersections. It can be seen that intersection 2 has the lowest MSE. Since intersection 2 is located in the center of the main road, the flow data of this intersection is also related to the traffic conditions of intersection 1 and intersection 3. Intersection 1 and intersection 3 are located at the boundary of the main road. There is only one upstream or downstream intersection, which is less affected. Therefore,

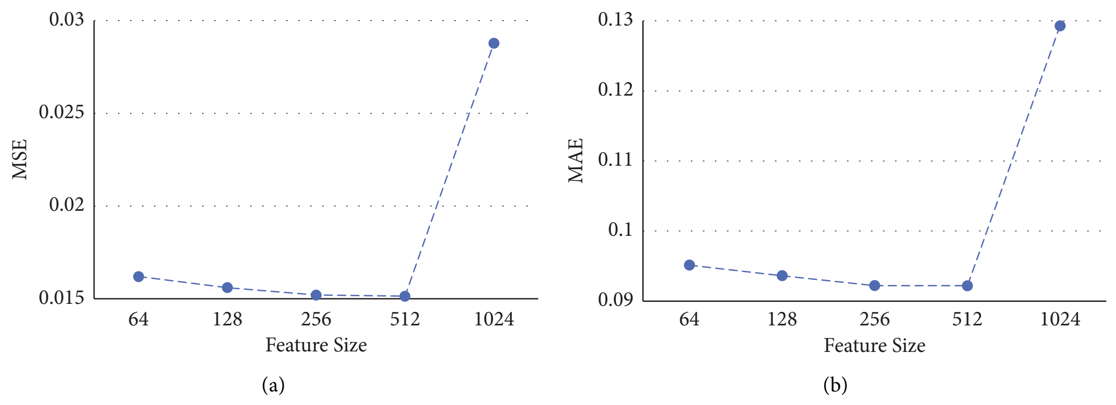


Figure 7: Effect of a hidden layer's dimension on the effectiveness of the task of predicting traffic flow at related intersections. (a) MSE. (b) MAE.

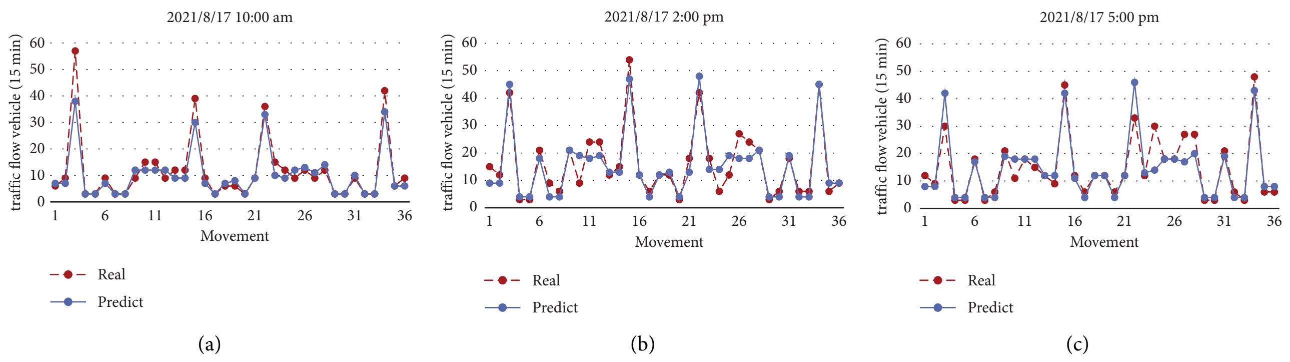


Figure 8: The comparison between the real traffic volume and the traffic volume predicted by CNNformer$^+$ at a single time step, i.e., 10 a.m., 2 p.m., and 5 p.m. Each time step includes $36 (12 \times 3)$ traffic movements. (a) CNNformer$^+$ at 10 a.m. (b) CNNformer$^+$ at 2 p.m. (c) CNNformer$^+$ at 5 p.m.

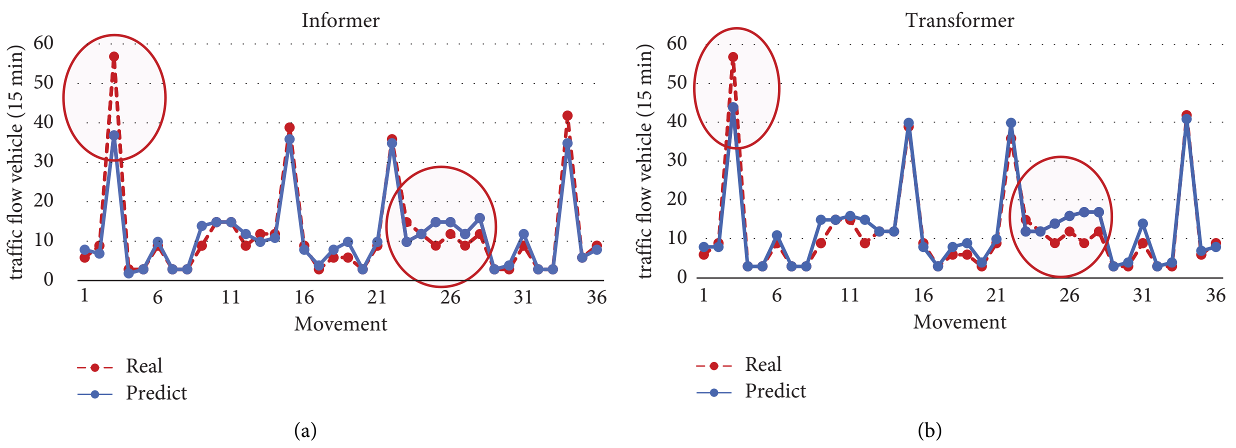


Figure 9: The comparison of real traffic volume at 10 a.m. with traffic volume predicted by informer and transformer. Each time step includes $36 (12 \times 3)$ traffic movements. (a) Informer at 10 a.m. (b) Transformer at 10 a.m.

Table 3: Volume prediction results for each intersection.

Intersections	MSE
Intersection 1	0.01483
Intersection 2	0.01563
Intersection 3	0.01500

the change in traffic volume is more regular, reducing the difficulty of prediction.

Figure 10 shows the forecast results of traffic volume in different time step sizes. It can be seen that MSE and MAE also begin to decrease significantly with the increase of time step size. This is because the transformer requires a large

Table 4: Volume predication results for each movement of intersection 2.

Movement of intersection 2	MSE
West -> North	0.01387
West -> South	0.01154
West -> East	0.01067
North -> West	0.01974
North -> South	0.01824
North -> East	0.01812
South -> West	0.01797
South -> North	0.02021
South -> East	0.01972
East -> West	0.01276
East -> North	0.01341
East -> South	0.01137

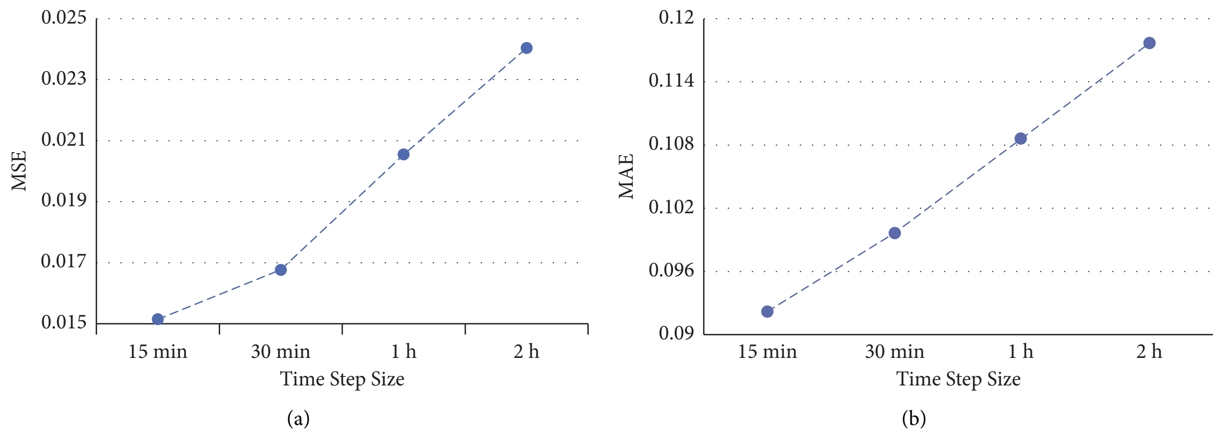


Figure 10: Volume prediction results using different time step sizes. (a) MSE. (b) MAE.

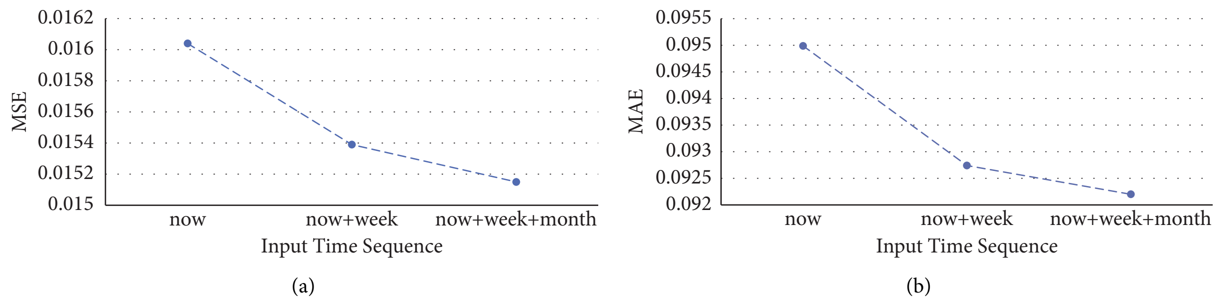


Figure 11: Volume prediction results using different time sequences. (a) MSE. (b) MAE.

amount of data for training. With the increase of time step size, the number of input samples of the model decreases, and the number of learned traffic volume features decreases, this increases the difficulty of prediction. The results show that a small time step size should be selected as far as possible in traffic flow prediction.

Figure 11 shows the influence of different sequences on prediction accuracy. "now" refers to the input only using the traffic flow data of the previous time window $(X_{t-H}, X_{t-H+1}, \ldots, X_t)$. "now + week" refers to the input contains the traffic flow data of the previous time window and the simultaneous data of the week before the previous time window $(X_{t-H-\text{week}}, X_{t-H-\text{week}+1}, \ldots, X_{t-\text{week}})$. "now + week + month" refers to the input contains the traffic flow data of the previous time window, the simultaneous data of the week before the previous time window, and the simultaneous data of the previous month in the previous time window $(X_{t-H-\text{month}}, X_{t-H-\text{month}+1}, \ldots, X_{t-\text{month}})$. The findings demonstrate that the minimal MSE and MAE are reached by taking into account all three time windows.

5. Conclusion

Transformer has advantages in dealing with time series tasks. Many current research works are based on the transformer architecture to establish models for various series tasks and have achieved good results beyond the traditional models in many application fields. To tackle the problem of traffic safety oriented multi-intersection flow prediction, in this research, a new architecture integrating CNN and transformer is proposed from the viewpoint of accuracy improvement, making it more suitable for the traffic flow prediction task of associated intersections. The comparative experiment with informer and other baseline models proves the superiority of the new architecture.

In the research work of this paper, the following results have been achieved:

(i) A new intersection traffic flow prediction model CNNformer $^+$ is proposed, which considers that the traffic flow data at the associated intersection is a group of spatiotemporal sequences, using CNN to extract the spatial features of the data can significantly improve the prediction accuracy of the transformer model.

(ii) The average pooling layer successfully learns the periodicity of the traffic flow data, increasing the model's forecast accuracy. Experiments on the simulated network dataset demonstrate the superiority of the proposed method.

Data Availability

The data used to support the findings of the study are available on request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

Tingting Fu and Peng Liu contribute to problem formulation and solution design. Qianwen Yu contributes to the implementation of the algorithms and writing. Haksrun Lao is responsible for the simulation experiment and part of the writing. Shaohua Wan helps revise the paper.

Acknowledgments

This work was supported by the Natural Science Foundation of China under Grant no. 62172134.

References

[1] L. Cheng, J. Liu, G. Xu et al., "SCTSC: a semicentralized traffic signal control mode with attribute-based blockchain in IoVs," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1373–1385, 2019.

[2] A. H. Abdul Hanan, M. Yazid Idris, O. Kaiwartya, M. Prasad, and R. Ratn Shah, "Real traffic-data based evaluation of vehicular traffic environment and state-of-the-art with future issues in location-centric data dissemination for vanets," *Digital Communications and Networks*, vol. 3, no. 3, pp. 195–210, 2017.

[3] S. Liu, J. Yu, X. Deng, and S. Wan, "Fedcpf: an efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1616–1629, 2022.

[4] W. Wei, R. Yang, H. Gu, W. Zhao, C. Chen, and S. Wan, "Multi-objective optimization for resource allocation in vehicular cloud computing networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25536–25545, 2022.

[5] C. Chen, L. Liu, S. Wan, X. Hui, and Q. Pei, "Data dissemination for industry 4.0 applications in Internet of vehicles based on short-term traffic prediction," *ACM Transactions on Internet Technology*, vol. 22, no. 1, pp. 1–18, 2021.

[6] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: theoretical basis and empirical results," *Journal of Transportation Engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[7] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, "Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and levenberg–marquardt algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 644–654, 2012.

[8] M. Castro-Neto, Y. S. Jeong, M. K. Jeong, and L. D. Han, "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6164–6173, 2009.

[9] T. Alladi, V. Kohli, V. Chamola, and F. R. Yu, "A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems," *Digital Communications And Networks*, vol. 8, 2022, https://www.sciencedirect.com/science/article/pii/S2352864822001407.

[10] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[11] S. Alkheder, W. Alkhamees, R. Almutairi, and M. Alkhedher, "Bayesian combined neural network for traffic volume short-term forecasting at adjacent intersections," *Neural Computing & Applications*, vol. 33, no. 6, pp. 1785–1836, 2020.

[12] W. Alajali, W. Zhou, and W. Sheng, "Traffic flow prediction for road intersection safety," in *Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Guangzhou, China, October 2018.

[13] K. Tan, D. Bremner, J. Le Kernec, L. Zhang, and M. Imran, "Machine learning in vehicular networking: an overview," *Digital Communications and Networks*, vol. 8, no. 1, pp. 18–24, 2022.

[14] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 1–13, 2020.

[15] Y. Tu, S. Lin, J. Qiao, and B. Liu, "Deep traffic congestion prediction model based on road segment grouping," *Applied Intelligence*, vol. 51, no. 11, pp. 8519–8541, 2021.

[16] Y. Ren, D. Zhao, D. Luo, H. Ma, and P. Duan, "Global-local temporal convolutional network for traffic flow prediction,"

IEEE Transactions on Intelligent Transportation Systems, vol. 23, pp. 1–7, 2020.

[17] C. Ma, G. Dai, and J. Zhou, “Short-term traffic flow prediction for urban road sections based on time series analysis and LSTM_BILSTM method,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 1–10, 2021.

[18] B. Du, H. Peng, S. Wang et al., “Deep irregular convolutional residual LSTM for urban traffic passenger flows prediction,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2019.

[19] C. Chen, Y. Zhang, Z. Wang, S. Wan, and Q. Pei, “Distributed computation offloading method based on deep reinforcement learning in ICV,” *Applied Soft Computing*, vol. 103, no. 7, pp. 107-108, 2021.

[20] W. Qu, J. Li, L. Yang et al., “Short-term intersection traffic flow forecasting,” *Sustainability*, vol. 12, no. 19, p. 8158, 2020.

[21] D. Kim and O. Jeong, “Cooperative traffic signal control with traffic flow prediction in multi-intersection,” *Sensors*, vol. 20, no. 1, p. 137, 2019.

[22] W. Li, X. J. Ban, J. Zheng, H. X. Liu, C. Gong, and Y. Li, “Real-time movement-based traffic volume prediction at signalized intersections,” *Journal of Transportation Engineering Part A Systems*, vol. 146, no. 8, Article ID 40, 2020.

[23] J. Guo, M. Bilal, Y. Qiu, C. Qian, X. Xu, and K.-K. Raymond Choo, “Survey on digital twins for internet of vehicles: fundamentals, challenges, and opportunities,” *Digital Communications And Networks*, vol. 12, 2022, https://www.sciencedirect.com/science/article/pii/S235286482200116X.

[24] B. Fan, Z. Su, Y. Chen, Y. Wu, C. Xu, and T. Q. S. Quek, “Ubiquitous control over heterogeneous vehicles: a digital twin empowered edge AI approach,” *IEEE Wireless Communications*, pp. 1–8, 2022.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *Computer Science*, vol. 1, 2013.

[26] H. Zhou, S. Zhang, J. Peng, S. Zhang, and W. Zhang, “Informer: beyond efficient transformer for long sequence time-series forecasting,” 2020, https://arxiv.org/abs/2012.07436.