NetTimeFormer: An Easily Deployable Network Traffic Prediction Model*

Licheng Zhou[1,†]

[1]Wuhan Fiberhome Technical Services Co.,Ltd. , Wuhan 430068, China

Abstract

Network traffic prediction plays a critical role in network management and optimization. While traditional deep learning models, such as recurrent neural networks and convolutional neural networks, perform well in time series prediction, they still face some challenges in network traffic prediction. Firstly, these models are prone to information loss when dealing with long time dependencies. Second, these models tend to have high complexity and are difficult to operate effectively in real-world deployments. To address these issues, we propose an improved lightweight transformer model. The model effectively captures long-term dependencies by introducing a self-attention mechanism, and achieves the goal of lightweight by modifying the shape of the embedding module and the computation of the self-attention score, making it more suitable for practical deployment. Preliminary experimental results show that our improved transformer model outperforms existing methods in terms of both prediction accuracy and efficiency.

Keywords

Deep learning, Artificial intelligence, Network traffic prediction, lightweight, Attention mechanism

1. Introduction

With the popularity of the Internet and advances in network technology, network size continues to expand and network services and applications become more diverse. Network traffic can reflect user activities and assess network load and operational status [1,2]. By predicting network traffic, network operation can be managed based on complex characteristics and changing rules, identifying bottlenecks, potential threats and failures, optimizing configuration, intrusion detection and fault management. As a result, network traffic prediction has become a hot research topic [3].

With the rapid proliferation of Internet of Things (IoT) devices and the complexity of network environments, predicting network traffic is becoming increasingly important to ensure network performance, optimize resource allocation and enforce security. However, with these technological advances comes the proliferation of edge devices, which typically have limited computing power and storage resources [4,5]. However, to meet the demand for highly accurate network traffic prediction, existing research relies on complex deep learning models and large-scale data processing algorithms that perform well when running on cloud servers, but face serious challenges when applied to edge devices [6].

Many existing predictive models require significant computing resources, including not only powerful central processing units (CPUs) and graphics processing units (GPUs), but also large amounts of memory and storage. These requirements exceed the processing power of most edge devices, making it expensive and difficult to run such models on these devices. In addition, the power constraints of edge devices also mean that highly loaded computational tasks cannot be sustained for long periods of time, further limiting the practical application of these models [7]. Therefore, the key issue in current research is how to design and optimize network traffic

The Second International Conference of Young Scientists on Artificial Intelligence for Sustainable Development (YAISD), May 8-9, 2025, Ternopil, Ukraine

* Corresponding author.

† These authors contributed equally.

15926470282@163.com (Z. Licheng)
0009-0008-3676-034X (Z. Licheng)

prediction models to reduce the consumption of computational resources and adapt to the processing power of edge devices, while maintaining high prediction accuracy. Meanwhile, effective extraction and representation of information is crucial in network traffic prediction and deep learning models. However, traditional models often suffer from information loss or ignore important features when dealing with long sequence data, especially when dealing with complex multi-dimensional data [8].

In comparison with traditional deep learning-based models and other machine learning algorithms, the Transformer model demonstrates robust global feature extraction and long-range feature modelling capabilities. Consequently, it represents a research priority for forecasting future time series. The attention mechanism allows the model to capture pertinent information in a more flexible manner by dynamically adjusting the extent of the model's attention to different components of the input data, thereby preventing the loss of crucial features during the transfer of information [9,10]. In particular, the attention mechanism is capable of adaptively assigning disparate weights to each time step or feature in accordance with the contextual information inherent to the input sequence. This process ensures that the model not only focuses on local information but also effectively focuses on the global context when dealing with long sequences or multi-dimensional data, thereby substantially improving the completeness of information retention and the accuracy of prediction [11].

The application of traditional self-attention mechanisms to long sequence data presents a significant computational resource consumption challenge, despite the excellent performance observed in the capture of global contextual information and the improvement of model performance. The computational complexity of the self-attention mechanism is typically proportional to the square of the sequence length. Consequently, the demand for computational resources increases exponentially when dealing with large-scale or high-dimensional data[12,13]. In particular, the self-attention mechanism necessitates the computation of a similarity matrix for each element in the sequence with all other elements. This process not only requires a significant amount of memory but also results in a considerable increase in the computational burden. This is particularly problematic when high real-time performance is required or when running on resource-constrained edge devices. The high computational and memory consumption inherent to self-attention mechanisms presents a significant obstacle to their wide deployment in practical applications, particularly in the context of ultra-long sequences or large-scale datasets. In such cases, limitations in computational resources may lead to suboptimal performance or even the inability to run the model at all[14].

In order to address the aforementioned challenges, this study employs convolutional neural networks (CNNs) in conjunction with self-attention mechanisms to introduce an inductive bias, with the objective of reducing the reliance on the traditional embedding module in response to the amount of input data. The NetTimeFormer model employs multi-scale convolutional coding in the embedding module, thereby replacing the input coding module and position coding module of the standard Transformer. This configuration enables the model to consider the global feature extraction capacity while acquiring an inductive bias, which mitigates the impact of long time series information loss. Furthermore, the conventional self-attention mechanism is modified by adopting a linear attention operating paradigm, which serves to further reduce the model's computational resource consumption [15].

The main contributions of this paper are as follows:

1. To address the issue of data loss during transmission, this study employs multi-scale convolutional coding, replacing the input coding module and position coding module of the standard Transformer. This improvement guarantees the resilience of the information in the presence of varying lengths of long-time series samples within a flow.
2. By enhancing the attention mechanism of the conventional Transformer, the computational complexity is reduced to a linear scale. The enhanced attention mechanism markedly diminishes the number of parameters and the computational burden, thereby considerably reducing the deployment cost in authentic engineering contexts.
3. The enhanced Transformer model, designated NetTimeFormer, was developed in the present study. Evaluation of NetTimeFormer on two publicly accessible datasets indicates that it demonstrates remarkable performance and minimal computational resource consumption.

2. Methods

2.1. CNN-based embedding module

In this study, in order to more effectively capture and retain the key information in the flow of long time series and to address the potential loss of information during transmission, we propose a multi-scale convolutional coding strategy as an alternative to the input coding and position coding modules in the standard Transformer. This enhanced design ensures the robustness and consistency of information when dealing with traffic long time series samples of varying lengths, and significantly enhances the feature extraction capability of the model. The improved embedded module is shown in Figure 1

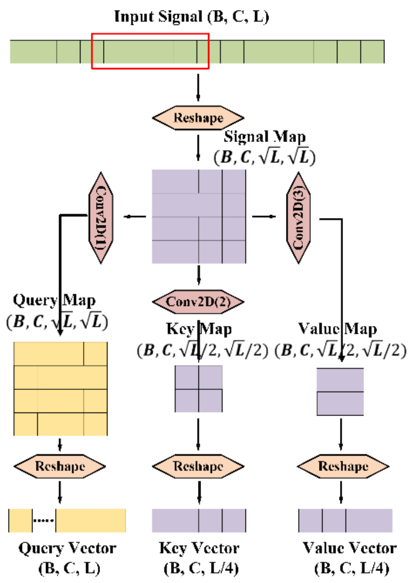

Figure 1: Example figure caption

Specifically, assume that the input data are vectors of length L with shape(B, C, L), where B is the batch size, C is the number of channels, and LLL is the sequence length. In order to convert this dimensional data into a format suitable for 2D convolution operation, we first reorganize (reshape) the input data to obtain a four-dimensional tensor of shape $\left(B, C, \sqrt{L}, \sqrt{L}\right)$. This operation can be expressed as:

$$X_{reshape} = Reshape(X) \tag{1}$$

where X is the original input and $X_{reshape}$ is the reorganized input.

Subsequently, three independent convolutional neural networks (CNNs) were devised for the generation of query, key and value vector representations (denoted as Q, K and K respectively). The process of forward propagation is as follows:

$$\begin{aligned} Q_{map} &= \sigma\left(Conv_{q,2}\left(Normal\left(Vonv_{q,1}\left(X_{reshape}\right)\right)\right)\right) \\ K_{map} &= \sigma\left(Conv_{k,2}\left(Normal\left(Vonv_{k,1}\left(X_{reshape}\right)\right)\right)\right) \\ V_{map} &= \sigma\left(Conv_{v,2}\left(Normal\left(Vonv_{v,1}\left(X_{reshape}\right)\right)\right)\right) \end{aligned} \tag{2}$$

Where $Conv_{i,1}(\cdot)$ represents the first convolution kernel of the i-vector. $Normal(\cdot)$represents the batch normalisation operation. $Conv_{i,1}(\cdot)$ represents the second convolution kernel of the i-vector, which is a dot convolution. $\sigma(\cdot)$ represents the activation function.

Finally, these 2D feature mappings are again transformed into a 1D sequence representation suitable for processing by the self-attention mechanism as:

$$Q, K, V = Flatten(Q_{map}, K_{map}, V_{map}) \tag{3}$$

The introduction of a multi-scale convolutional embedding module enables the effective extraction of global features from input sequences, while also enhancing the model's capacity to perceive features at varying time scales through the fusion of multi-scale information. This design ensures the robustness and accuracy of the model in the task of long-term flow prediction while capturing essential features.

2.2. Attention Mechanisms for Linear Complexity

In order to overcome the computational resource consumption problem of the traditional self-attention mechanism in long time series processing, and at the same time improve the feature extraction capability of the embedding module in different time scales, we introduce a new model architecture based on the linear attention mechanism and the multiscale convolutional embedding module[16,17]. The linear attention mechanism reduces the traditional $O(L^2)$ to $O(L)$ by optimizing the complexity of the attention weight computation, which significantly reduces the consumption of computational resources and is suitable for longer time series data processing. The calculation process is shown in Figure 2:

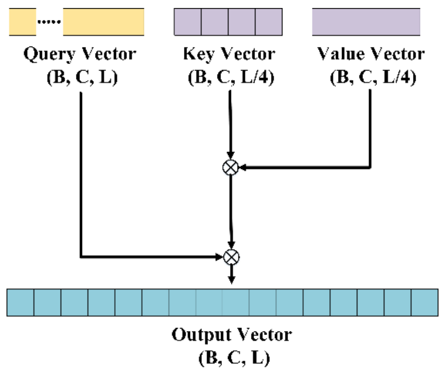

Figure 2: Improved Attention Computing Module

In the design of our proposed method, the input signal is subjected to multiple convolution operations to generate three different feature vectors corresponding to the query ($Q \in R^{B \times C \times L}$), key ($K \in R^{B \times C \times L}$), and value ($V \in R^{B \times C \times L}$).

In the case of the traditional self-attention mechanism, the dot product of the query and key is calculated in order to obtain the attention score matrix.

To avoid the problem of vanishing or exploding gradients due to too large a dot product value, it is common to divide the dot product result by $\sqrt{d_k}$ Perform scaling. The formula is expressed as follows:

$$\text{Attention Scores} = \frac{QK^\top}{\sqrt{d_k}} \tag{4}$$

Then, the Softmax function is applied to the scaled attention score matrix to obtain the relevance weights of each query with respect to all keys:

$$\text{AttentionWeights} = Softmax\left(\frac{QK^\top}{\sqrt{d_k}}\right) \tag{5}$$

Where the Softmax operation is applied row by row to ensure that each query has a weight sum of 1 with all keys.

In traditional self-attention mechanisms, the computational complexity is usually $O(L^2)$, where L is the length of the input sequence. This is because when computing the attention weights, the Q and K of the dot product, generating a sequence of size $L \times L$ correlation matrix. However, for the processing of long time series signals, the consumption of computational resources will increase as L increases significantly, so it is crucial to reduce the computational complexity.

In the improved method proposed in this paper, we adopt a linear attention mechanism to significantly reduce the computational complexity. We pair keys K and the value of V The transpose is matrix multiplied to generate the correlation matrix A.

$$A = K^{\top} \cdot V \tag{6}$$

where $A \in R^{B \times C \times C}$.

Next, by computing the query A and the matrix A are multiplied to obtain the final time-domain attention weights Q:

$$Q' = Q^{\top} \cdot A \tag{7}$$

Since $Q \in R^{B \times L \times C}$, so that $Q' \in R^{B \times L \times C}$.Since in long time series signals, usually $L \gg C$, using this linear attention mechanism increases the computational complexity from the traditional $O(L^2)$ is reduced to $O(L)$. This greatly reduces the consumption of computational resources and allows the method to process long time-span sensor signals more efficiently.

2.3. Lightweight improvements to output modules

The function of the resultant output layer is to integrate the channel feature vectors of feature mapping and provide the final prediction vector. However, previous research has often neglected the in-depth study and improvement of classifiers. A traditional predictor consists of a multilayer perceptron (MLP) consisting of two fully connected layers (fc). The number of neurons in the last FC is the length of the predicted sequence. It is calculated as follows:

$$y = W_{fc2} \cdot \mathrm{GELU}\left(\mathrm{BN}\left(W_{fc1} \cdot \mathrm{X}\right)\right) \tag{8}$$

where W_{fc1} and W_{fc2} denote the weights of the two FC layers, respectively, ignoring the bias terms. BN(·) denotes batch normalization. GELU(·) is the activation function.

It has been demonstrated that increasing the width of the hidden layer improves the representation of the model, thereby enhancing its effectiveness in capturing complex patterns and structures in the input data. However, an increase in width also entails a higher computational cost and may result in model overfitting with respect to the training data. To address these issues, in this study we employ a grouped MLP to redesign the classifier. This approach can effectively balance the expressive power and computational efficiency of the model while maintaining its performance, reducing the risk of overfitting and enabling more flexible adaptation to the requirements of different tasks.

Suppose the input to the backbone module is $x \in R^{B \times C \times \mathrm{L}}$, then this input data is divided into k non-overlapping subgroups, i.e. $x_i \in R^{B \times \mathrm{C} \times \frac{L}{k}}$, where $i = 1,2,\dots,k$. Subsequently, for each of the subgroups we x_i Independent linear transformations are performed. The width of the hidden layer of the classifier is fixed to twice the number of input neurons and the final output dimension is the fault type d of the number of faults. For each subgroup i, its linear transformation can be expressed as follows:

$$L(x_i) = W_i x_i^T + b_i^T \tag{9}$$

where x_i^T denotes the transpose of the input. b_i^T denotes the bias term. W_i denotes the weight matrix for the grouped linear transformation. Then the output is:

$$Y_H = W_{out}\left(\mathrm{GELU}\left(\mathrm{BN}\left(\mathrm{Concat}\left(L(x_1), L(x_2), \dots, L(x_k)\right)\right)\right)\right) \tag{10}$$

When the bias term is ignored, a classifier constructed using a traditional multilayer perceptron (MLP) produces a total number of parameters of $L \times 2L + 2L \times D = 2N(N + D)$. And when dividing the input data into k groups, the classifier constructed using the grouped MLP strategy produces a total number of parameters of $\frac{k \times 2L}{k} \times k + 2L \times D = 2L\left(\frac{L}{k} + D\right)$. Therefore, our design is able to reduce $L\left(\frac{k-1}{k}\right)$ of the number of parameters.

2.4. Overall structure

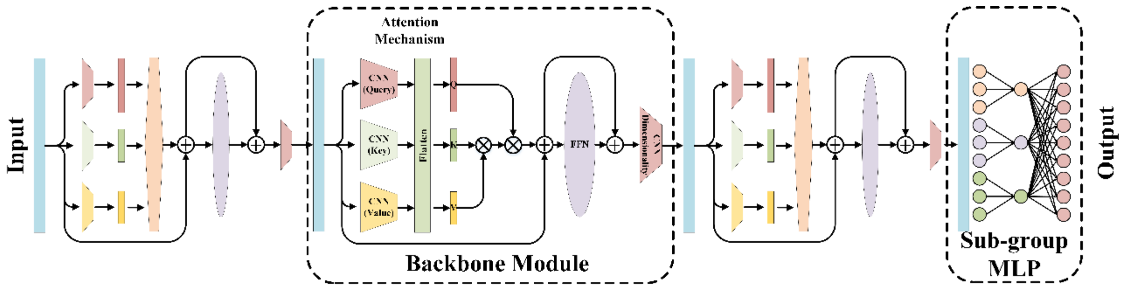

Figure 3: Overall structure

NetTimeFormer consists of three phases including a backbone module for attention computation and a grouped MLP that outputs prediction results. The overall structure is shown in Figure 3. Detailed structural information is shown in Table 1. where N is the length of the prediction sequence and the default value is 96.

Table 1
NetTimeFormer network structure table

Layer	Input	Output
Input Sequences	1×96	4×96
$Backbone_1$	1×96	4×96
$Backbone_2$	4×96	8×96
$Backbone_3$	8×96	16×96
Predictor Output	16×96	1×N
Parameters		5664
MFLOPs		0.16

3. Experiments

3.1. Datasets

In order to validate the sequence prediction accuracy of the model in real scenarios, two distinct datasets have been employed. The initial dataset was gathered from the core network of a European city by a private Internet Service Provider (ISP), encompassing the core network regions of 11 major European cities. The dataset provides a detailed account of internet traffic on the transatlantic link between 06:57 on 7 June 2005 and 11:17 on 31 July 2005, with data collected at five-minute intervals. This dataset provides insight into internet transmissions between multiple European cities, offering a valuable perspective on cross-border network traffic. The second dataset is derived from the UK academic backbone and provides a comprehensive overview of the overall traffic patterns within the UK academic network. The dataset records traffic from 09:30 on 19 November 2004 to 11:11 on 27 January 2005, with data collected at five-minute intervals. This dataset offers a comprehensive insight into the overall traffic patterns and trends within the UK academic network. The combination of these two datasets provides a multi-level perspective for analyzing internet traffic behavior, encompassing both inter-city traffic between major European cities and the overall traffic profile of the academic network. This makes them a valuable resource for research, as they offer a representative overview of internet traffic patterns.

The dataset can be downloaded from the link https://github.com/xiaohuiduan/network-traffic-dataset.

3.2. Evaluation metrics

In network traffic prediction tasks, we usually use a variety of evaluation metrics in order to assess the prediction performance of models. In this paper, three commonly used metrics, Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE), are chosen to quantify the prediction accuracy of the model.

Mean Square Error (MSE) is a measure of the average of the squared error between the predicted value and the true value, and is an indicator that is sensitive to large errors. the smaller the MSE, the higher the predictive accuracy of the model. The formula is as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2 \tag{11}$$

where n is the number of samples. y_i is the first i actual value of $\widehat{y_i}$ is the first i predicted value. The MSE reflects the extent to which the predicted value deviates from the true value and is sensitive to outliers due to the presence of squares.

The Mean Absolute Error (MAE) is the average of the absolute values of all prediction errors and provides a direct measure of prediction error. Unlike MSE, MAE is less sensitive to large errors. The formula is as follows:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}| \tag{12}$$

where $|\cdot|$ denotes an absolute value. The MAE reflects the average degree to which the model deviates from the true value across all predicted values and has better robustness because it is not overly sensitive to outliers.

The Mean Absolute Percentage Error (MAPE) is the average of the prediction error as a percentage of the true value and is used as a measure of relative error. The units of the MAPE are in per cent, making it more interpretable. The formula is as follows:

$$\mathrm{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - \widehat{y_i}|}{y_i} \tag{13}$$

MAPE provides a relative measure of prediction error and is suitable for comparing data of different magnitudes. However, MAPE has a significant effect on the true value y_i close to zero produces unstable results and therefore needs to be used with caution in some cases.

3.3. Experimental results

In order to ensure the fairness and credibility of the experiments, all experiments used the same setup, and the optimizer used Adam. the gradient was computed using the MSE loss function, and the cosine annealing was used to learn the rate scheduling algorithm, which is: $\mathrm{Lr}(t) = \mathrm{Lr}_{\min} + \frac{1}{2}(\mathrm{Lr}_{\max} - \mathrm{Lr}_{\min})\left(1 + \cos\left(\frac{t}{T_{\max}}\pi\right)\right)$, where max_lr and lrmin denote the maximum and minimum values of the learning rate, respectively. In the experiments, the chosen batch size = 64, $\mathrm{Lr}_{\max} = 10\mathrm{e}-3$, $\mathrm{Lr}_{\min} = 10\mathrm{e}-4$, epoch = 200.

As shown in Table 2, we perform experiments related to network traffic prediction using NetTimeFormer. The experiments are compared with the traditional Transformer with the advanced sequence prediction model FEDformer. The results show that our model obtains the best experimental accuracy under each prediction sequence. And it can still maintain a low error in long sequence prediction. Under EC dataset, NetTimeFormer improves 10-20% compared to FEDformer. The ISP dataset has a smoother waveform than EC dataset, so the time series model can achieve higher accuracy. NetTimeFormer shows excellent prediction accuracy under ISP dataset. The MSE is only 0.049 at a prediction length of 128.

Table 2

Prediction accuracy of the model at different prediction lengths when the input sequence length is 96.

Models		NetTimeFormer			FEDformer			Transformer		
Metric		MSE	MAE	MAPE	MSE	MAE	MAPE	MSE	MAE	MAPE
EC	**48**	0.034	0.143	10.1	0.034	0.143	10.1	0.064	0.143	10.1
	96	0.049	0.151	10.3	0.076	0.179	11.1	0.088	0.201	13.4
	128	0.060	0.163	10.7	0.089	0.202	12.7	0.101	0.294	14.1
ISP	**48**	0.013	0.097	5.4	0.031	0.117	7.1	0.044	0.012	9.8
	96	0.027	0.112	7.8	0.055	0.157	10.1	0.076	0.189	11.4
	128	0.049	0.151	9.1	0.069	0.188	12.2	0.097	0.246	13.1

A visual presentation of the sequence prediction results is shown in Figure 4. It can be clearly seen that NetTimeFormer's prediction results fit very well. The model captures the details of the fluctuations in the sequence.

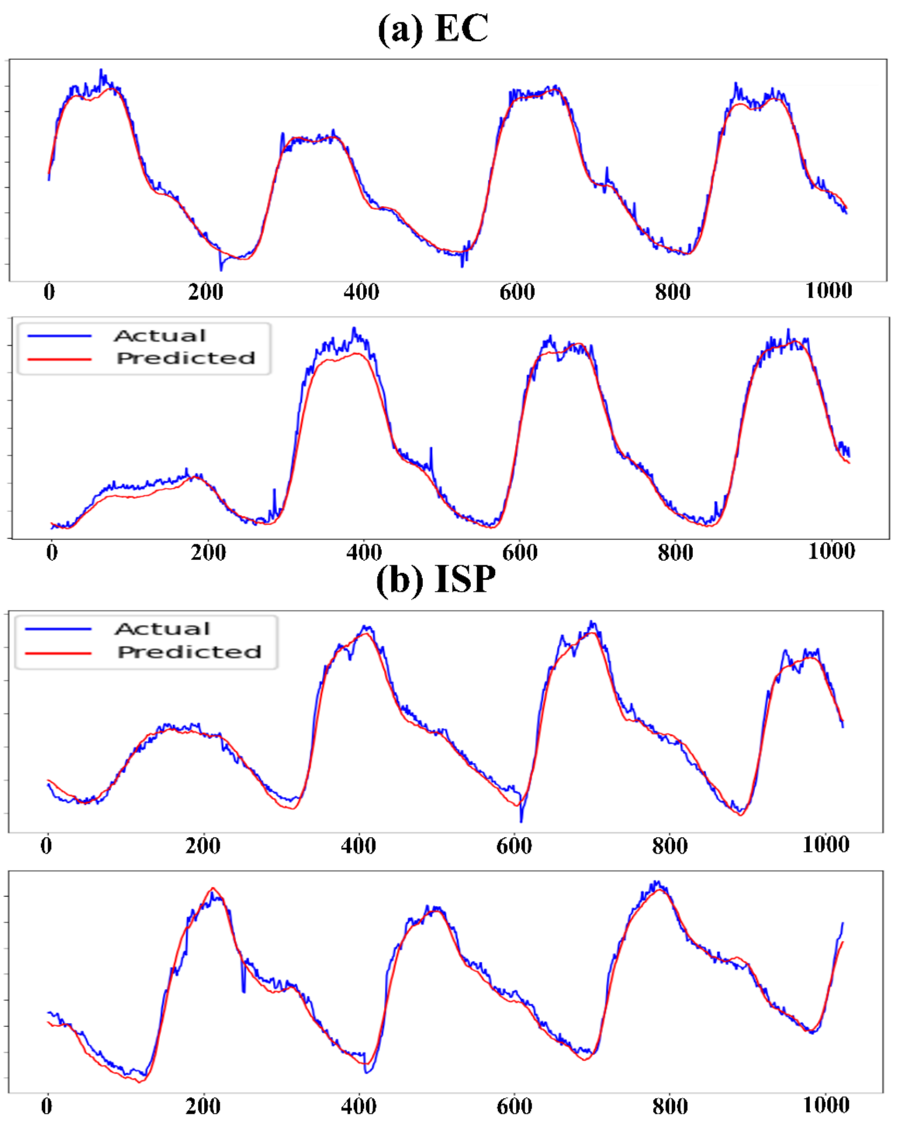

Figure 4: Demonstration of NetTimeFormer's actual traffic prediction

4. Conclusion

In this paper, we propose an improved lightweight Transformer model, NetTimeFormer, to address the problems of long time-dependent information loss and high computational complexity faced by traditional deep learning models in network traffic prediction. We effectively maintain the integrity of long time series information by introducing multi-scale convolutional coding to replace the input coding and location coding modules of the standard Transformer. In addition, the optimized self-attention mechanism reduces the computational complexity to a linear level, which significantly reduces the number of parameters and computational burden of the model, and lowers the actual deployment cost. Experimental results on two publicly available datasets show that NetTimeFormer excels in prediction accuracy and computational efficiency, significantly outperforming existing methods, especially on resource-constrained edge devices. In summary, this study is not only innovative in model design, but also experimentally verifies its practicality and excellent performance in network traffic prediction, which provides valuable references for further research and practical applications in related fields.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Tian H, Guo K. Chaotic characteristic analysis and prediction of bottleneck-delay time series under the Internet macro-topology. The European Physical Journal Plus. 2024 Jun 1;139(6):494.

[2] Joshi M, Hadi TH. A review of network traffic analysis and prediction techniques. arXiv preprint arXiv:1507.05722. 2015 Jul 21.

[3] Vinayakumar R, Soman KP, Poornachandran P. Applying deep learning approaches for network traffic prediction. In2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) 2017 Sep 13 (pp. 2353-2358). IEEE.

[4] Feng H, Shu Y. Study on network traffic prediction techniques. InProceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005. 2005 Sep 26 (Vol. 2, pp. 1041-1044). IEEE.

[5] Ferreira GO, Ravazzi C, Dabbene F, Calafiore GC, Fiore M. Forecasting network traffic: A survey and tutorial with open-source comparative evaluation. IEEE Access. 2023 Jan 11;11:6018-44.

[6] Sanon SP, Reddy R, Lipps C, Schotten HD. Secure federated learning: An evaluation of homomorphic encrypted network traffic prediction. In2023 IEEE 20th Consumer Communications & Networking Conference (CCNC) 2023 Jan 8 (pp. 1-6). IEEE.

[7] Alkanhel R, El-kenawy ES, Elsheweikh DL, Abdelhamid AA, Ibrahim A, Khafaga DS. Metaheuristic Optimization of Time Series Models for Predicting Networks Traffic. CMC-COMPUTERS MATERIALS & CONTINUA. 2023 Jan 1;75(1):427-42.

[8] Li F, Feng J, Yan H, Jin G, Yang F, Sun F, Jin D, Li Y. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. ACM Transactions on Knowledge Discovery from Data. 2023 Feb 20;17(1):1-21.

[9] Lai J, Chen Z, Zhu J, Ma W, Gan L, Xie S, Li G. Deep learning based traffic prediction method for digital twin network. Cognitive Computation. 2023 Sep;15(5):1748-66.

[10] Khan A, Fouda MM, Do DT, Almaleh A, Rahman AU. Short-term traffic prediction using deep learning long short-term memory: Taxonomy, applications, challenges, and future trends. IEEE Access. 2023 Aug 29.

[11] Bao B, Yang H, Yao Q, Guan L, Zhang J, Cheriet M. Resource allocation with edge-cloud collaborative traffic prediction in integrated radio and optical networks. IEEE Access. 2023 Jan 16;11:7067-77.

[12] Chen J, Xu M, Xu W, Li D, Peng W, Xu H. A flow feedback traffic prediction based on visual quantified features. IEEE Transactions on Intelligent Transportation Systems. 2023 May 9;24(9):10067-75.

[13] Xu Y, Cai X, Wang E, Liu W, Yang Y, Yang F. Dynamic traffic correlations based spatio-temporal graph convolutional network for urban traffic prediction. Information Sciences. 2023 Apr 1;621:580-95.

[14] Xu Y, Cai X, Wang E, Liu W, Yang Y, Yang F. Dynamic traffic correlations based spatio-temporal graph convolutional network for urban traffic prediction. Information Sciences. 2023 Apr 1;621:580-95.

[15] Liu S, Feng X, Ren Y, Jiang H, Yu H. DCENet: A dynamic correlation evolve network for short-term traffic prediction. Physica A: Statistical Mechanics and its Applications. 2023 Mar 15;614:128525.

[16] Fang H, Deng J, Bai Y, Feng B, Li S, Shao S, Chen D. CLFormer: A lightweight transformer based on convolutional embedding and linear self-attention with strong robustness for bearing fault diagnosis under limited sample conditions. IEEE Transactions on Instrumentation and Measurement. 2021 Dec 3;71:1-8.

[17] Han D, Pan X, Han Y, Song S, Huang G. Flatten transformer: Vision transformer using focused linear attention. InProceedings of the IEEE/CVF international conference on computer vision 2023 (pp. 5961-5971).