Supplemental Material to

LightCts: A Lightweight Framework for Correlated Time Series Forecasting

1 TIME AND SPACE COMPLEXITY ANALYSIS OF S/T-OPERATORS

1.1 Key Notations

First, we recap the key notations related to *correlated time series* (CTS) in Table 1.

Table 1. Notation

Notation	Description
X	An indexed set of correlated time series (CTS)
H	Embedded CTS feature maps
$\mathtt{N}$	Number of time series in X
$\mathtt{T}$	Number of time steps in X
$\mathtt{F}$	Number of features in X
$\mathtt{D}$	Embedding size of S/T-operators
$\mathtt{P}$	Number of historical time steps used in CTS forecasting

Notably, we suppose that the input CTS data $X \in \mathbb{R}^{\mathtt{N}\times\mathtt{P}\times\mathtt{F}}$ has been embedded into $H \in \mathbb{R}^{\mathtt{N}\times\mathtt{P}\times\mathtt{D}}$ by the input and embedding module. Consequently, the input shape of S/T-operators is $(\mathtt{N}\times\mathtt{P}\times\mathtt{D})$.

The detailed complexity analyses for T-operators and S-operators are presented in Sections 1.2 and 1.3, respectively.

1.2 T-operator

1.2.1 CNN family. In this section, we present the detailed compositions of T-operators from the CNN family, and specify their time and space complexities. We take the TCN (Temporal Convolutional Network) as the representative [2]. The main difference between the TCN and the standard CNN is the convolution order such that the TCN should ensure the convolution output is based on the information from previous time steps. Nevertheless, the convolution order does not influence the time and space complexities.

With the raw input CTS $X \in \mathbb{R}^{\mathtt{N}\times\mathtt{P}\times\mathtt{F}}$ being embedded into the latent representation $H \in \mathbb{R}^{\mathtt{N}\times\mathtt{P}\times\mathtt{D}}$ by the embedding module, a TCN layer cast on H is formalized as follows.

$$\mathrm{TCN}(H \mid \delta, \mathtt{K}) = H', \text{ where}$$
$$H'[i;t;d] = \sum_{k=0}^{\mathtt{K}-1} \left(H[i;t-\delta\times k;:] \cdot \mathrm{W}^d[k;:]\right)$$

is the d-th ($d \in [0,\mathtt{D})$) output feature map at time step t ($t \in [0,\mathtt{P})$) of the i-th ($i \in [0,\mathtt{N})$) time series, $\mathrm{W}^d \in \mathbb{R}^{\mathtt{D}\times\mathtt{K}}$ is the d-th convolutional filter, and $\mathtt{K}$ (often as small as 2 or 3) and δ are the kernel size and dilation rate, respectively.

Specifically, we adopt $\mathtt{D}$ filters, each of which is in size of $(\mathtt{K}\times\mathtt{D})$. Notably, the filter's kernel size $\mathtt{K}$ is usually a small value (e.g., 2 or 3) that can be regarded as a constant. Since all parameters of TCNs come from convolution filters, the **space complexity** is $O(\mathtt{D}^2)$ (i.e., $\mathtt{D}$ filters each taking $O(\mathtt{D})$ parameters). Notably, since features from different time steps and nodes typically share one set of convolution filters [13, 14, 16], the space complexity is not related to $\mathtt{N}$ and $\mathtt{P}$.

Besides, for one filter in one convolution calculation, its time complexity is $O(\mathtt{D})$ since the number of its input channels (i.e., embedding size) is $\mathtt{D}$[1] With $\mathtt{P}$ convolution operations along the temporal dimension for $\mathtt{N}$ CTS nodes using $\mathtt{D}$ filters, the overall **time complexity** is $O(\mathtt{D}^2\cdot\mathtt{N}\cdot\mathtt{P})$.

[1] As explained above, we also disregard the kernel size $\mathtt{K}$ in the time complexity analysis.

1.2.2 RNN family. We proceed to detail the compositions of RNN-based T-operators and derive their time and space complexities. There are mainly two RNN-based T-operators, namely LSTMs (Long Short-Term Memory) [8] and GRUs (Gate Recurrent Units) [5]. Their main difference is the number of gates, which is a constant that does not affect the complexities. In the following, we take the LSTM as the example.

Equation 1 presents the workflow of an LSTM operator in a particular time step t:

$$\begin{aligned} i_t &= \sigma\left(\mathrm{W}_{ii} H(i)_t + \mathrm{W}_{hi} h_{t-1}\right), \\ f_t &= \sigma\left(\mathrm{W}_{if} H(i)_t + \mathrm{W}_{hf} h_{t-1}\right), \\ g_t &= \tanh\left(\mathrm{W}_{ig} H(i)_t + \mathrm{W}_{hg} h_{t-1}\right), \\ o_t &= \sigma\left(\mathrm{W}_{io} H(i)_t + \mathrm{W}_{ho} h_{t-1}\right), \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t, \\ h_t &= o_t \odot \tanh\left(c_t\right), \end{aligned} \tag{1}$$

where $H(i)_t$ is the i-th time series features at the t-th time step of H, h_t is the hidden state at the t-th time step of LSTM, i_t, f_t, g_t, and o_t denote the outputs of the input gate, the forget gate, the cell update, and the output gate, respectively. The computation of the four components i_t, f_t, g_t, and o_t are the same in terms of time complexity. Thus, we take the input gate i_t as an example for time complexity analysis:

$$i_t = \sigma\left(\mathrm{W}_{ii} X(i)_t + \mathrm{W}_{hi} h_{t-1}\right) \tag{2}$$

where $\mathrm{W}_{ii} \in \mathbb{R}^{\mathsf{H_h} \times \mathsf{D}}$ and $\mathrm{W}_{hi} \in \mathbb{R}^{\mathsf{H_h} \times \mathsf{H_h}}$ are the weight matrices of the input gate, which determine the value of the input, $\mathsf{H_h}$ is the size of the hidden state.

Since the LSTM is made up of these gates which consist of two weight matrices with shapes of $\mathsf{H_h} \times \mathsf{D}$ and $\mathsf{H_h} \times \mathsf{H_h}$, the space complexity of an LSTM T-operator is $O(\mathsf{H_h D} + \mathsf{H_h}^2)$. For simplification, we substitute $\mathsf{H_h}$ with D as they are normally similar in values for practical implementations. As a result, the **space complexity** is simplified as $O(\mathsf{D}^2)$.

Similarly, since features from different time steps and nodes typically share one set of gates [1, 3, 4, 9], the space complexity is not related to N and P.

On the other hand, Equation 2 consists of the computations of $\mathrm{W}_{ii} H(i)_t$ and $\mathrm{W}_{hi} h_{t-1}$. Thus, the time complexity of Equation 2 is $O(\mathsf{H_h}(\mathsf{H_h} + \mathsf{D}))$, simplified as $O(\mathsf{D}^2)$. Considering that we have N time series with the length of P, the overall **time complexity** of an LSTM operator is $O(\mathsf{D}^2\mathsf{NP})$.

1.2.3 Transformer family. A standard Transformer-based T-operator consists of an MHA layer and an FFN layer [11].

In particular, an **MHA layer** is a concatenation of a number h of repeated self-attention modules (heads) in parallel:

$$\mathrm{MHA}(H^{\mathrm{PE}}) = \mathrm{concat}(\{\mathrm{head}_i(H^{\mathrm{PE}})\}_{i=1}^{\mathsf{h}}) \tag{3}$$

$$\mathrm{head}_i(H^{\mathrm{PE}}) = \mathrm{softmax}\left(H^{\mathrm{I}}\right) \cdot V_i \tag{4}$$

$$H^{\mathrm{I}} = (Q_i \cdot K_i^{\mathsf{T}})/\sqrt{\mathsf{D/h}}, \tag{5}$$

$$Q_i = H^{\mathrm{PE}} \cdot \mathrm{W}_i^Q, K_i = H^{\mathrm{PE}} \cdot \mathrm{W}_i^K, V_i = \mathrm{H}^{\mathrm{PE}} \cdot \mathrm{W}_i^V. \tag{6}$$

where all learnable matrices $\mathrm{W}_i^* \in \mathbb{R}^{\mathsf{D} \times \frac{\mathsf{D}}{\mathsf{h}}}$ are used to convert the embedding size from D to $\mathsf{D/h}$. In Equation 4, each head $\mathrm{head}_i(\cdot)$ is a self-attention module that produces attention scores for its input $H^{\mathrm{PE}} \in \mathbb{R}^{\mathsf{N} \times \mathsf{D}}$ by making the input interact with itself. It should be noticed that, H^{PE} is the position embedding results of the input CTS feature H, of which computation only involves matrix addition calculations, not contributing to the time complexity.

An **FFN layer** merges the output $H^{\text{MHA}} \in \mathbb{R}^{\mathsf{P}\times\mathsf{D}}$ of Equation 3 and provides non-linear activations via two fully-connected layers:

$$\text{FFN}(H^{\text{MHA}}) = \text{FFN}^{(1)}(H^{\text{MHA}}) \cdot \text{W}_2 + \text{b}_2, \tag{7}$$

$$\text{FFN}^{(1)}(H^{\text{MHA}}) = \text{ReLU}(H^{\text{MHA}} \cdot \text{W}_1 + \text{b}_1), \tag{8}$$

where $\text{W}_1 \in \mathbb{R}^{\mathsf{D}\times\mathsf{D}'}$, $\text{W}_2 \in \mathbb{R}^{\mathsf{D}'\times\mathsf{D}}$, $\text{b}_1 \in \mathbb{R}^{\mathsf{D}'}$, and $\text{b}_2 \in \mathbb{R}^{\mathsf{D}}$ are learnable matrices and biases. The two layers first enlarge the embedding size from D to D', and then scale it back to D.

For one Transformer-based T-operator, it has $3 \times \mathsf{h}$ linear projection matrices $\text{W}_i^Q, \text{W}_i^K, \text{W}_i^V \in \mathbb{R}^{\mathsf{D}\times\mathsf{D}/\mathsf{h}}$ in the MHA, and two FFN layers with the weight matrices of $\mathbb{R}^{\mathsf{D}\times\mathsf{D}_\mathsf{F}}$ and $\mathbb{R}^{\mathsf{D}_\mathsf{F}\times\mathsf{D}}$, respectively, D_F is the embedding size of the first fully-connected layer of the FFN layer. Therefore, the overall **space complexity** is $O(3 \times \mathsf{h} \times \mathsf{D} \times \mathsf{D}/\mathsf{h} + 2 \times \mathsf{D} \times \mathsf{D}_\mathsf{F})$, which equals to $O(\mathsf{D}^2)$. It should be also noticed that, as the most studies do [6, 10, 12, 15, 17], the features from all time steps and nodes share one set of Transformer parameters, thus, the parameter number is unrelated to N and P.

As for the time complexity, considering $3 \times \mathsf{h}$ linear projection matrices with the shape of $\mathbb{R}^{\mathsf{D}\times\mathsf{D}/\mathsf{h}}$ in the MHA, and we need to exploit the temporal features of N nodes, the time complexity of this part is $O(3\times\mathsf{N}\times\mathsf{h}\times\mathsf{P}\times\mathsf{D}\times\mathsf{D}/\mathsf{h})$, which can be simplified as $O(\mathsf{D}^2\mathsf{NP})$. Since we conduct self-attention operations on the basis of time steps, according to the computation processes of Equation 4 and Equation 5, the time complexity of this part is $O(\mathsf{N} \times \mathsf{h} \times \mathsf{P} \times \mathsf{P} \times \mathsf{D}/\mathsf{h} + \mathsf{N} \times \mathsf{h} \times \mathsf{P} \times \mathsf{P} \times \mathsf{D}/\mathsf{h})$, which can be simplified as $O(\mathsf{P}^2\mathsf{DN})$. For the FFN layer, since it consists of two linear layers with D_F and D filters, the time complexity of this part is $O(2 \times \mathsf{N} \times \mathsf{P} \times \mathsf{D} \times \mathsf{D}_\mathsf{F})$, which is $O(\mathsf{D}^2\mathsf{NP})$. In summary, the total **time complexity** is $O(\mathsf{D}^2\mathsf{NP} + \mathsf{P}^2\mathsf{DN})$, which equals to $O(\mathsf{NDP}(\mathsf{P} + \mathsf{D}))$.

1.3 S-operator

1.3.1 GCN family. Each GCN S-operator performs the following two steps: neighborhood information aggregation and nonlinear transformation. The neighborhood information aggregation is performed by the matrix multiplication between the spatial adjacency matrix and the input node features, followed by a nonlinear transformation performed by convolution operation over the aggregated feature. In particular, a GCN S-operator conducted one GCN operation on one time step slice $H_G \in \mathbb{R}^{\mathsf{N}\times\mathsf{D}}$ is shown as follows:

$$GCN(H_G) = \sigma(A \cdot H_G \cdot W) \tag{9}$$

where $\mathsf{A} \in \mathbb{R}^{\mathsf{N}\times\mathsf{N}}$ is a normalized adjacency matrix representing the spatial correlations, $W \in \mathbb{R}^{\mathsf{D}\times\mathsf{D}}$ is the GCN weight matrix, and $\sigma(\cdot)$ is the nonlinear activation function.

The **space complexity** derives from the GCN weight matrix $W \in \mathbb{R}^{\mathsf{D}\times\mathsf{D}}$, which is $O(\mathsf{D}^2)$, and is also unrelated to N and P. This is because the features from all time steps and nodes also share the same GCN weight matrix [7, 9, 12, 14].

Besides, from Equation 9, the time complexity for one time step slice is calculated as $O(\mathsf{N}^2\mathsf{D}+\mathsf{ND}^2)$. Considering P time steps, the **time complexity** can be expressed as $O(\mathsf{NDP}(\mathsf{N} + \mathsf{D}))$.

1.3.2 Transformer family. Similar to Transformer-based T-operators, a standard Transformer-based S-operator also consists of an MHA layer and an FFN layer and adopts the same computation mechanism, except for the slicing method of the input features. For S-operators, since we focus on exploiting the correlations among the information of each node within P time steps, the input features for the self-attention are partitioned into N slices of $\mathsf{P} \times \mathsf{D}$ features, which are P slices of $\mathsf{N} \times \mathsf{D}$ features in Transformer-based T-operators.

For one Transformer-based S-operator, it also has $3 \times \mathsf{h}$ linear projection matrices $\text{W}_i^Q, \text{W}_i^K, \text{W}_i^V \in \mathbb{R}^{\mathsf{D}\times\mathsf{D}/\mathsf{h}}$ in the MHA, and two FFN layers with the weight matrices of $\mathbb{R}^{\mathsf{D}\times\mathsf{D}_\mathsf{F}}$ and $\mathbb{R}^{\mathsf{D}_\mathsf{F}\times\mathsf{D}}$, respectively. Therefore, the overall **space complexity** is also $O(3 \times \mathsf{h} \times \mathsf{D} \times \mathsf{D}/\mathsf{h} + 2 \times \mathsf{D} \times \mathsf{D}_\mathsf{F})$, which equals to

$O(D^2)$. Similarly, as most studies do [6, 10, 12, 15, 17], the features from all time steps and nodes share one set of Transformer parameters, thus, the parameter number is unrelated to N and P.

As for the time complexity, considering $3 \times h$ linear projection matrices with the shape of $\mathbb{R}^{D \times D/h}$ in the MHA, and we need to exploit the spatial correlations of P time steps, the time complexity of this part is $O(3 \times P \times h \times N \times D \times D/h)$, which can be simplified as $O(D^2NP)$. Since we conduct self-attention operations on the basis of nodes, like the computation processes of Equation 4 and Equation 5, the time complexity of this part is $O(P \times h \times N \times N \times D/h + P \times h \times N \times N \times D/h)$, which can be simplified as $O(N^2DP)$. For the FFN layer, since it consists of two linear layers with D_F and D filters, the time complexity of this part is $O(2 \times N \times P \times D \times D_F)$, which is $O(D^2NP)$.

Overall, the total **time complexity** is $O(D^2NP + N^2DP)$, which equals to $O(NDP(N + D))$.

2 PARAMETER STUDIES ON THE OTHER DATASETS

The experiments in this section are the supplement to those in Section 5.3.1 to Section 5.3.3 (on Page 9 to 11) in the submission.

2.1 Multi-step Forecasting

We perform parameter studies on the other three multi-step datasets, i.e. PEMS04, METR-LA, and PEMS-BAY. The results are shown in Fig. 1, Fig. 2, and Fig. 3.

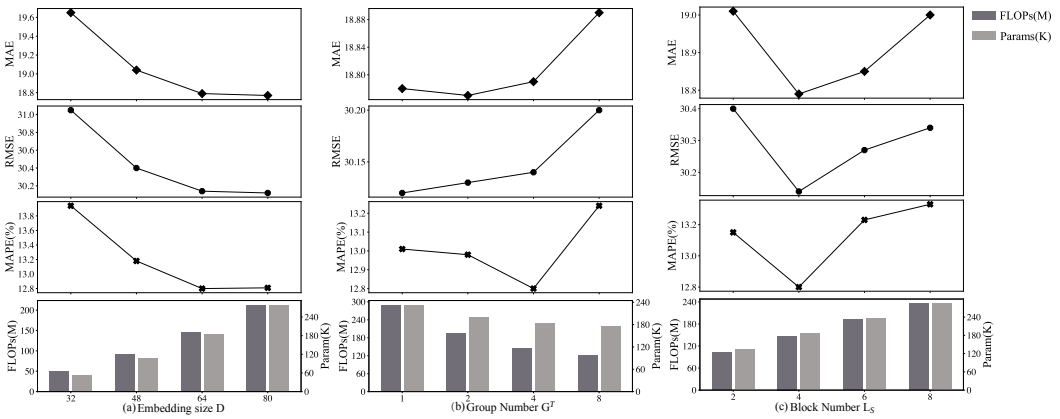

Fig. 1. Impact of (a) embedding size D, (b) group number G^T, and (c) attention block number L_S in GL-Former for multi-step forecasting on the PEMS04 dataset.

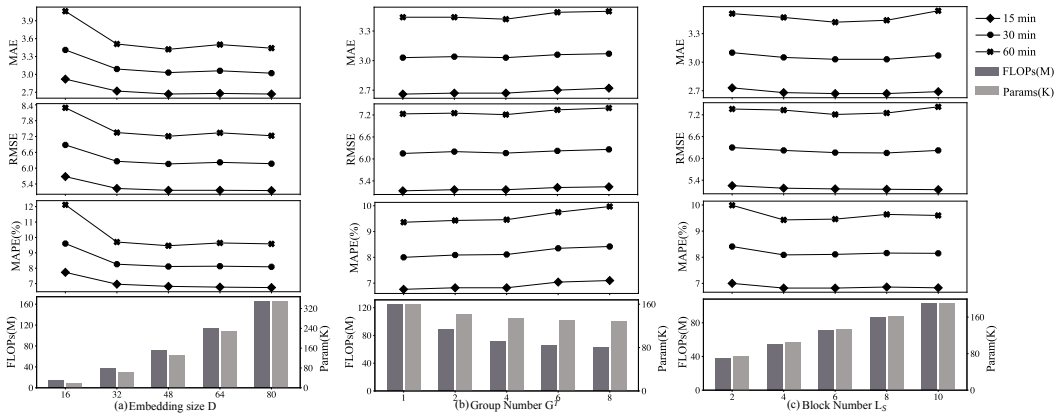

Fig. 2. Impact of (a) embedding size D, (b) group number G^T, and (c) attention block number L_S in GL-Former for multi-step forecasting on the METR-LA dataset.

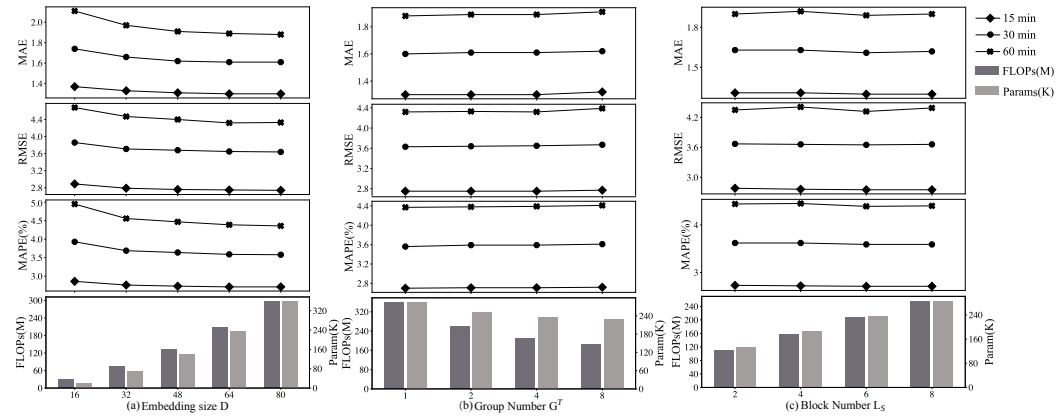

Fig. 3. Impact of (a) embedding size D, (b) group number G^T, and (c) attention block number L_S in GL-Former for multi-step forecasting on the PEMS-BAY dataset.

2.2 Single-step Forecasting

We also perform parameter studies on the other single-step dataset, i.e. Electricity. The result is shown in Fig. 4.

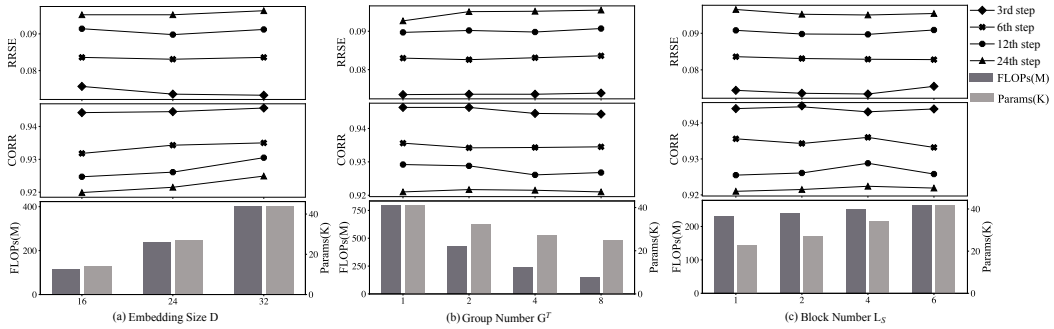

Fig. 4. Impact of (a) embedding size D, (b) group number G^T, and (c) attention block number L_S in GL-Former for single-step forecasting on the Electricity dataset.

3 PRELIMINARY EXPERIMENTS OF THE IMPACTS OF G^M AND G^F

The experiments in this section are the supplement to those in Section 5.3.2 (on Page 10 to 11) in the submission.

3.1 Multi-step Forecasting

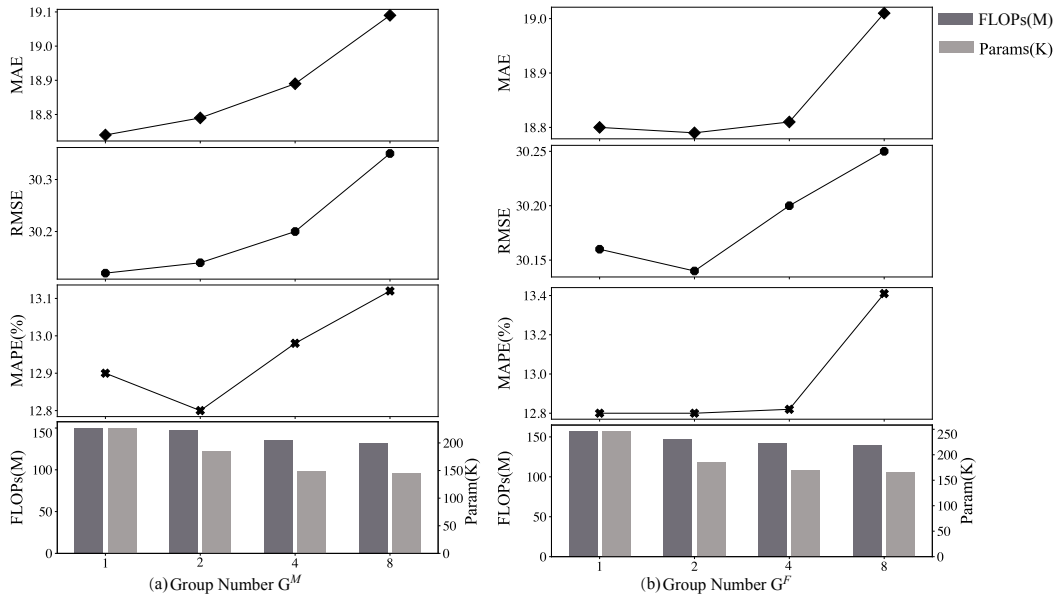

Fig. 5. Impact of (a) group number G^M and (b) group number G^F for multi-step forecasting on the PEMS04 dataset.

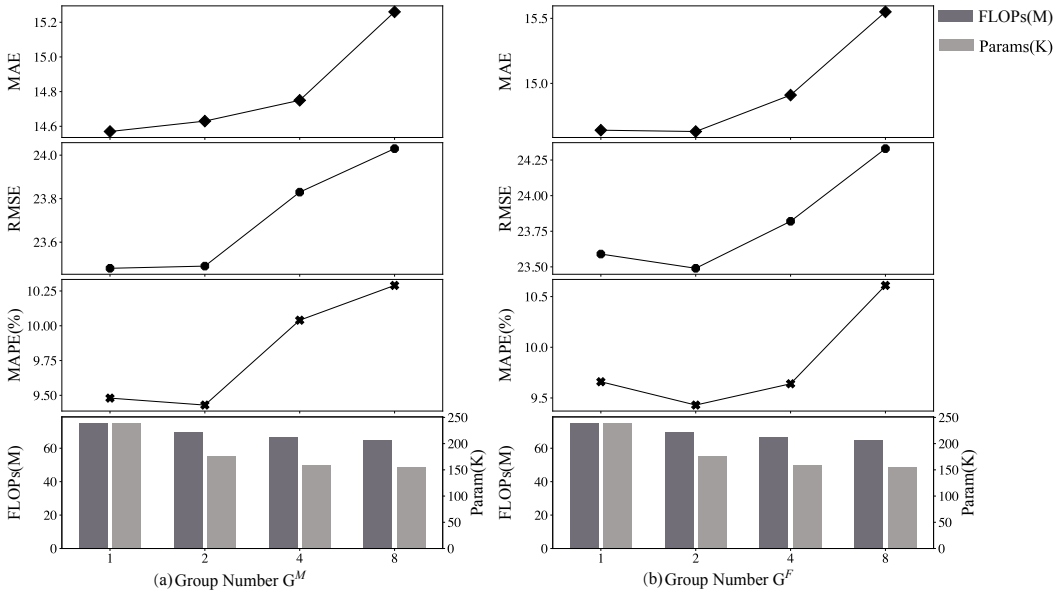

Fig. 6. Impact of (a) group number G^M and (b) group number G^F for multi-step forecasting on the PEMS08 dataset.

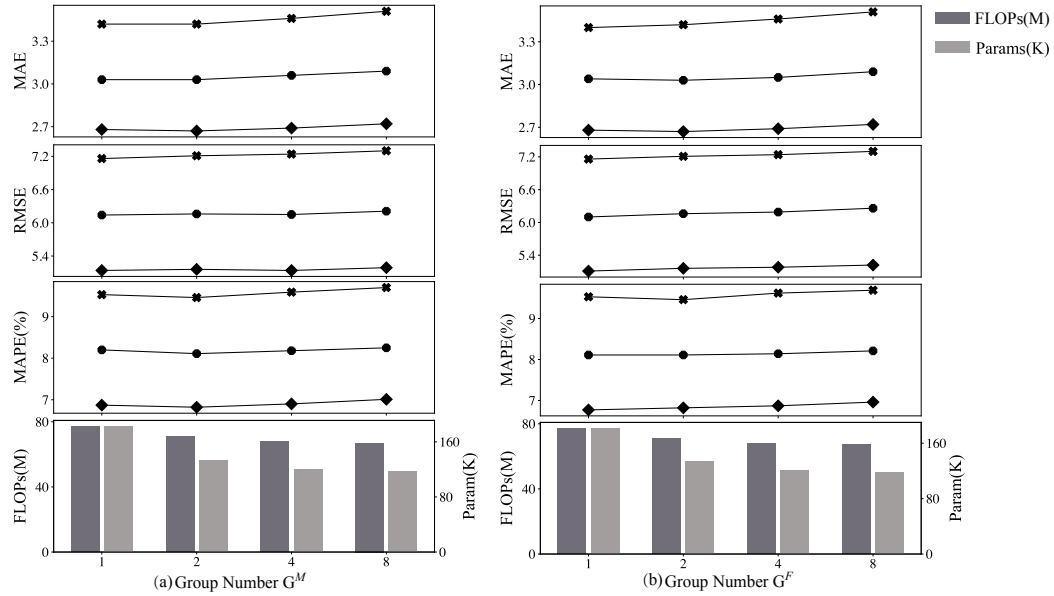

Fig. 7. Impact of (a) group number G^M and (b) group number G^F for multi-step forecasting on the METR-LA dataset.

3.2 Single-step Forecasting

4 ABLATION STUDIES ON THE OTHER DATASETS

The experiments in this section are the supplement to those in Section 5.4 (on Page 11 to 12) in the submission.

REFERENCES

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *NeurIPS*, pages 17804–17815, 2020.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[3] Yen-Yu Chang, Fan-Yun Sun, Yueh-Hua Wu, and Shou-De Lin. A memory-network based solution for multivariate time-series forecasting. *arXiv preprint arXiv:1809.02105*, 2018.

[4] Weiqi Chen, Ling Chen, Yu Xie, Wei Cao, Yusong Gao, and Xiaojie Feng. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In *AAAI*, pages 3529–3536, 2020.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] Jake Grigsby, Zhe Wang, and Yanjun Qi. Long-range transformers for dynamic spatiotemporal forecasting. *arXiv preprint arXiv:2109.12218*, 2021.

[7] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, pages 922–929, 2019.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, pages 1–10, 2018.

[10] Cheonbok Park, Chunggi Lee, Hyojin Bahng, Yunwon Tae, Seungmin Jin, Kihwan Kim, Sungahn Ko, and Jaegul Choo. ST-GRAT: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed. In *CIKM*, pages 1215–1224, 2020.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, volume 30, 2017.

[12] Xinle Wu, Dalin Zhang, Chenjuan Guo, Chaoyang He, Bin Yang, and Christian S Jensen. AutoCTS: Automated correlated time series forecasting. *PVLDB*, 15(4):971–983, 2021.

[13] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *KDD*, pages 753–763, 2020.

[14] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, pages 1907–1913, 2019.

[15] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. Spatial-temporal transformer networks for traffic flow forecasting. *arXiv preprint arXiv:2001.02908*, 2020.

[16] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *IJCAI*, pages 3634–3640, 2018.

[17] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, pages 11106–11115, 2021.