Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 6446941, 14 pages
https://doi.org/10.1155/2022/6446941

*Research Article*

# A Sparse Gating Convolutional Recurrent Network for Traffic Flow Prediction

**Xiaohui Huang**,[1] **Jie Tang**,[1] **Zhiying Peng**,[1] **Zhiyi Chen**,[2] **and Hui Zeng**[1]

[1]*School of Information Engineering, East China Jiaotong University, Nanchang 330013, China*
[2]*Jiangxi Human Resources and Social Security Publicity Center, Beijing, China*

Correspondence should be addressed to Xiaohui Huang; hxh016@gmail.com

Received 13 October 2021; Revised 6 January 2022; Accepted 26 January 2022; Published 22 February 2022

Academic Editor: Rahib Abiyev

With the continuous development of deep learning, more and more huge deep learning models are developed by researchers, which leads to an exponential increase of the parameters of models. Therein, the convolutional recurrent network as a type of widely used deep learning method is often employed to handle spatiotemporal data, e.g., traffic data. However, because of the large number of parameters in the model, the convolutional recurrent network needs to consume a lot of computing resources and time in the training process. To reduce the consumption of resources, we propose a sparse convolutional recurrent network with a sparse gating mechanism that is able to reduce the complexity of the network by an improved gate unit while keeping the performance of the model. We evaluate the performance of our proposed network on traffic flow datasets, and the experimental results show that the parameters of the model are significantly reduced under the condition of similar prediction accuracy compared with the traditional convolutional recurrent network.

## 1. Introduction

Deep learning [1] has achieved great success in various fields, e.g., computer vision [2, 3] and natural language processing [4], and it has also been introduced into the field of traffic flow prediction by some scholars [5–7]. Meanwhile, with the development of the economy, the transportation network in recent years has also achieved rapid expansion. However, traffic congestion is becoming an increasingly serious issue. The traffic flow prediction of different key points in the transportation network is the basis for solving the issue. To improve the accuracy of traffic flow prediction, various complex deep networks are developed, which leads to an overwhelming increase of parameters and resource consumption. Convolutional long short-term memory (ConvLSTM) network was proposed by Shi et al. [8] for precipitation nowcasting, which introduced the convolutional operation into the fully connected LSTM [9]. Ballas et al. [10] proposed a convolutional gated recurrent unit (ConvGRU) for video representation, which uses a convolutional structure to capture spatial features. Then, they are applied to traffic flow prediction by many researchers. For example, Zonoozi et al. [11] proposed a ConvGRU method based on the periodic characteristics of the traffic flow data, which employs ConvGRU to extract the spatiotemporal representation of input data. Zang et al. [12] developed a deep learning approach for traffic speed prediction, which uses the ConvLSTM network to extract the temporal and spatial dependence of historical traffic data.

Since convolutional recurrent network has great advantages in extracting temporal and spatial information hidden in spatiotemporal data simultaneously, the method has been naturally applied to solve the prediction problem of traffic flow. However, the convolutional recurrent network has a large number of parameters and requires more resources in the training and testing process. Therefore, it is necessary to compress the convolutional recurrent network by reducing the amount of its parameters while the model is utilized to solve the prediction problem of traffic flow, especially on some resource-limited devices. In this paper, we propose a sparse convolutional recurrent network by using the sparse gate in ConvLSTM and ConvGRU to reduce the resource requirement. The contributions of this paper are summarized as follows:

(i) Based on ConvLSTM and ConvGRU, we develop four sparse convolutional recurrent networks: SConvLSTM, SConvLSTM+, SConvGRU, and SConvGRU+. The numbers of the parameters of our proposed networks are significantly reduced as opposed to their original versions (ConvLSTM and ConvGRU) on the basis of keeping the performance of the algorithms.

(ii) In the original gating unit of ConvGRU and ConvLSTM, we design a sparse gating mechanism, which reduces the parameters of the networks (ConvGRU and ConvLSTM) by reducing the input data of the gating unit.

(iii) We evaluate our proposed network on three real traffic flow datasets. Compared with the baseline model, our proposed network reduces the resource consumption and saves the training time under the condition of competitive prediction accuracy.

The outline of this paper is as follows: section 2 describes the related work of traffic flow prediction and sparse neural network. Then, the notation and problem definition are given in Section 3. Section 4 describes our proposed model (sparse convolutional recurrent network) in detail. In addition, Section 5 gives specific parameter settings and experimental analysis based on three real traffic flow datasets. Finally, in Section 6, we summarize the work of this paper and make an outlook for future work.

## 2. Related Work

*2.1. Traffic Flow Prediction.* In essence, traffic flow prediction is a time series forecasting problem, which predicts the future traffic value based on historical observations. Generally speaking, traffic flow forecasting methods can be divided into two categories: traditional prediction approaches and deep learning-based prediction approaches.

The first category is the traditional forecasting methods. HA [13] is a classic traffic flow prediction method, which uses the historical average as the prediction value of the next time interval. However, it can only estimate the traffic data of future time intervals and cannot capture the correlation between different time intervals. ARMA [14] prediction is a time series forecasting method for processing stationary series data, which builds the model to predict future values based on the autoregression (AR) model and the moving average (MA) model. ARIMA [15] is a traditional time series prediction approach used to deal with nonstationary series data. Firstly, the input data is converted into stationary series data by the $k^{\text{th}}$ order difference, and then the ARMA method is used to build a prediction model. The disadvantage of them is that they cannot capture the nonlinear characteristics of traffic data. VAR [16] was proposed by Sims et al. in 1980 for time series prediction, which uses the form of simultaneous equations to obtain the relationship between different traffic flows. In addition, the variant methods derived from them are SARIMA [17], SARIMAX [18], VARMA [19], and so on. The limitation of all these traditional methods is that it is difficult to capture the complex spatial and temporal features of traffic flow data because of the limited capacity of the model.

The second category is deep learning-based prediction approaches. Deep learning-based methods have a strong ability in dealing with nonlinear structural data and complex spatiotemporal series data. Therefore, these methods are introduced into predicting traffic flow by many researchers. Fu et al. [20] first employed the LSTM and GRU networks in the field of traffic flow forecasting, and the experimental results on the PeMS dataset prove that the prediction performance is better than the ARIMA model. To capture the hidden patterns of traffic flow data, Dai et al. [21] proposed a DeepTrend model based on a fully connected neural network and an LSTM network. Kang et al. [22] used the LSTM network to analyze the impact of different forms of traffic data on the prediction results, and the experimental results show that more external information is helpful to improve the performance of the model. Luo et al. [23] developed a deep learning method based on K-Nearest Neighbors (KNN) model [24] and the LSTM network. The experimental results on the real-time traffic flow dataset show that it achieves better performance than the existing prediction model. To capture the proximity, periodicity, and trend of traffic data, Wang et al. [25] proposed a deep learning model based on a convolutional recurrent network, which can effectively extract temporal dependence and spatial dependence. Chen et al. [26] developed a deep learning method based on Convolutional Neural Network (CNN) [27], LSTM network, and a fully connected neural network. The results show that the prediction accuracy is improved compared with LSTM and its variants. To improve the accuracy of taxi demand prediction, Li et al. [28] proposed a model based on deep learning, which uses a ConvLSTM network to capture spatiotemporal features.

*2.2. Sparse Method of Neural Network.* The deepening in the number of layers of the neural network brings about an increase in parameters and resource consumption. Therefore, it is necessary to compress the neural network under the premise of no significant performance degradation.

To improve the training speed and generalization ability of the network, Louizos et al. [29] developed a sparse method using $L_o$ regularization, which reduces the model complexity by pruning parameters. Liu et al. [30] compressed the model by reducing the number of redundant parameters. The experimental results prove that the speed of the training is effectively improved under the condition of minimizing the loss of accuracy. To improve computational performance and reduce the transmission of redundant data, Mukkara et al. [31] proposed a sparse convolutional neural network, which uses a strategy of pruning zero-valued weights to compress parameters. Dettmers et al. [32] proposed a sparse momentum to improve the training speed of deep neural networks. The experimental results on MNIST, CIFAR-10, and ImageNet datasets show that the sparse momentum achieves the most advanced sparse performance. To solve the problem that performance is limited by resources, Alford et al. [33] developed a pruning-based sparse neural network, which prunes the low weight of the trained densely connected network. Luo et al. [34] proposed a compression method for deep neural networks, which reduces the parameters of the network by pruning filters. In this paper, we develop a sparse gating mechanism for convolutional recurrent neural networks, which compresses the model by reducing the input data of the gating mechanism.

## 3. Notations and Problem Definition

*3.1. Grid Map.* As shown in Figure 1, we partition a city as an $M \times N$ grid map based on longitude and latitude [35], where a grid $(i.j)$ indicates that this area is located in the $i^{\text{th}}$ row and the $j^{\text{th}}$ column in the map.

*3.2. Inflow/Outflow Matrix.* Given the $t^{th}$ time interval, for the region $g(i,j)$, the inflow is defined as the sum of the crowd flow from other regions to the region $g(i,j)$ at the $t^{th}$ time interval. The outflow is defined as the sum of the crowd flow from the region $g(i,j)$ to other regions at the $t^{th}$ time interval. In this way, we can obtain the values of inflow and outflow of the region $g(i,j)$ at the $t^{th}$ time interval. Then, using the same method for all areas, the values of inflow and outflow of the $t^{th}$ time interval can be obtained [33]. Figure 2 shows the inflow and outflow of all areas in a certain time interval in Beijing. Therefore, the inflow and outflow matrices at any time interval can be expressed as a tensor. $X \in R^{2\times m\times n}$.

We use $\Gamma$ to represent all the crowd trajectories at the $t^{\text{th}}$ time interval $[s_t, e_t)$, and each trajectory is represented by a tuple. $\mathrm{Tr}\langle P_{\text{start}}, P_{\text{end}}, T_{\text{start}}, T_{\text{end}}\rangle$, which means that this trajectory starts at the region $P_{\text{start}}$ at time interval $T_{\text{start}}$, and ends at the region $P_{\text{end}}$ at time interval $T_{\text{end}}$. Then, the computing formula for inflow and outflow of the region $g(i,j)$ can be defined as follows:

$$\begin{aligned} x_{t,ij}^{\text{in}} &= \sum_{Tr\in\Gamma} \left| P_{\text{start}} \notin g(i,j) \wedge P_{\text{end}} \in g(i,j) \wedge T_{\text{start}} \in [s_t, e_t) \right|, \\ x_{t,ij}^{\text{out}} &= \sum_{Tr\in\Gamma} \left| P_{\text{start}} \in g(i,j) \wedge P_{\text{end}} \in g(i,j) \wedge T_{\text{start}} \notin [s_t, e_t) \right|, \end{aligned} \tag{1}$$

where $Tr$ is a trajectory. $x_{t,ij}^{\text{in}}/x_{t,ij}^{\text{out}}$ denotes the inflow/outflow in the region $g(i,j)$ at the $t^{th}$ time interval. $s_t$ represents the start time of the $t^{th}$ time interval. $e_t$ represents the end time of the $t^{th}$ time interval, and $|\cdot|$ denotes the cardinality of a set.

*3.3. Inflow/Outflow Prediction Problem.* As shown in Figure 3, traffic flow prediction is to predict the values of the traffic flow of the next $k$ time intervals based on a series of the historical data of the previous $l$ time intervals. The formula is as follows:

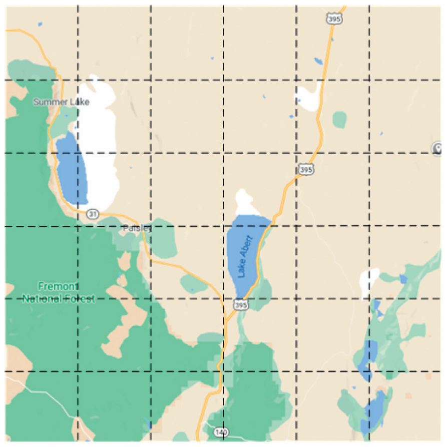


FIGURE 1: Grid-based map segmentation.

$$\{X_{t-l+1}, X_{t-l}, \ldots, X_t\} \xrightarrow{f} \{\widehat{X}_{t+1}, \widehat{X}_{t+2}, \ldots, \widehat{X}_{t+k}\}, \tag{2}$$

where $f$ denotes a flow prediction model, $\{X_{t-l+1}, \ldots, X_t\}$ represents the observation values of the previous $l$ time interval, and $\{\widehat{X}_{t+1}, \widehat{X}_{t+2}, \ldots, \widehat{X}_{t+k}\}$ represents the predicted values of $k$ time intervals in the future.

## 4. Sparse Convolutional Recurrent Neural Network

In this section, we introduce the details of the sparse convolutional recurrent neural network. There are two popular forms of convolutional recurrent neural networks: convolutional long short-term memory network and convolutional gated recurrent unit network. Based on the idea of eliminating redundant parameters in the sparse neural network methods, for convolutional long short-term memory network, we develop the sparse convolutional LSTM network (SConvLSTM) and SConvLSTM+. For the convolutional gated recurrent unit network, we develop the sparse convolutional GRU network (SConvGRU) and SConvGRU+.

*4.1. Inflow/Outflow Prediction Problem.* Figure 4(a) is the convolutional long short-term memory unit proposed by Shi et al. [8], which is composed of a main line part and some gating units. The input of the ConvLSTM unit is the cell state $C_{t-1}$ and the hidden state $H_{t-1}$ of the previous time interval, and the feature matrix $x_t$ of the current time interval. The output is the cell state $C_t$ and the hidden state $H_t$ of the current time interval. The main line part is mainly about the update of the cell state and hidden state, and the specific update equation is as follows:

$$\begin{aligned} \tilde{c} &= g(W_c \circledast x_t + U_c \circledast h_{t-1} + b_c), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}, \\ h_t &= o_t \odot g(c_t), \end{aligned} \tag{3}$$

where $\odot$ represents the Hadamard product. $\circledast$ represents the convolutional operation. $c_t$ is the updated equation about the cell state. $\tilde{c}$ is the updated equation about the input state.

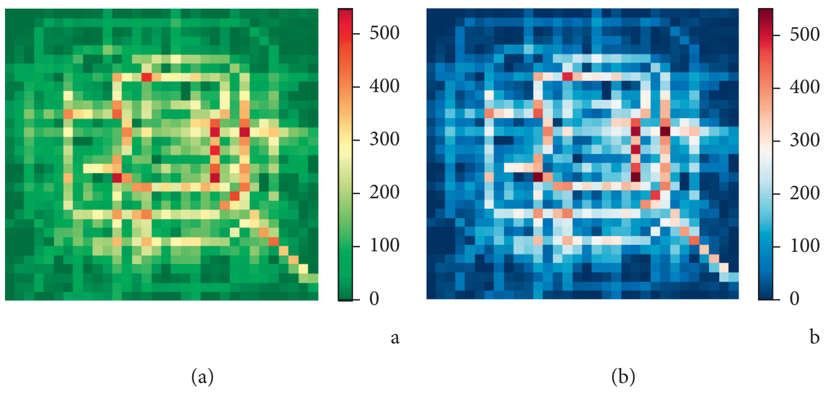


FIGURE 2: Examples of inflow matrix (a) and outflow matrix (b) in Beijing.

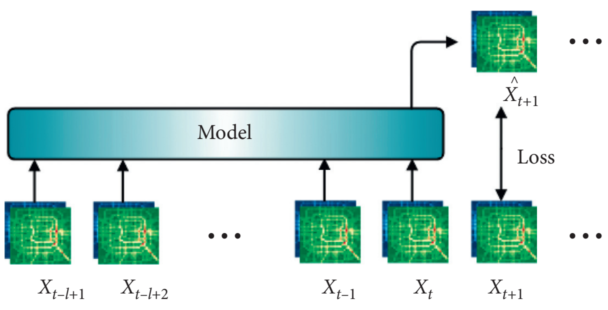


FIGURE 3: Traffic flow prediction.

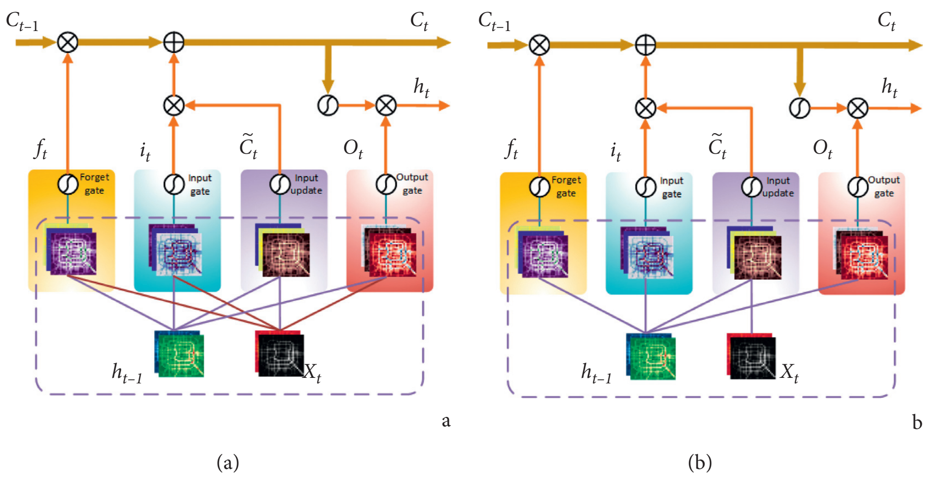


FIGURE 4: (a) ConvLSTM unit. (b) SConvLSTM unit.

$h_t$ is the updated equation about the hidden state. $g$ represents the Tanh function. $f_t, i_t, o_t$ represent the forget gate, the input gate, the output gate, respectively. $x_t$ denotes the input data of the $t^{\text{th}}$ time interval. $W_c, U_c, b_c$ are the learning parameters.

The gating unit is made of a forget gate, an input gate, and an output gate. The forget gate determines the degree of the forgetting of $C_{t-1}$. The input gate controls the degree of the input state to the cell state, and the output gate determines the degree of the cell state $C_t$ to the output state $h_t$. The gating mechanism of the ConvLSTM unit uses the hidden state $h_{t-1}$ of the previous time interval, the input $x_t$ of the current time interval, and the bias as input. Then, the updated formula of the gating mechanism is as follows:

$$\begin{aligned} i_t &= \sigma\left(W_i \circledast x_t + U_i \circledast h_{t-1} + b_i\right), \\ f_t &= \sigma\left(W_f \circledast x_t + U_f \circledast h_{t-1} + b_f\right), \\ o_t &= \sigma\left(W_o \circledast x_t + U_o \circledast h_{t-1} + b_o\right), \end{aligned} \tag{4}$$

where $\sigma$ denotes the Sigmoid function. $W, U, b$ represent the learning parameter.

The SConvLSTM unit is shown in Figure 4(b). We introduce a sparse gating mechanism to cut out the redundant parameters of the network. The main line part of the SConvLSTM unit is consistent with the ConvLSTM unit. The input of the gating mechanism in the ConvLSTM unit includes $H_{t-1}, x_t$ and the bias $b$, however, the input of the gating mechanism in the SConvLSTM unit is $H_{t-1}$ and the bias $b$. In this way, we can effectively reduce the complexity of the model while ensuring that $x_t$ is propagated to subsequent sequences in the input update formula $(\widetilde{c})$.

In Section 5, our experimental results on three traffic datasets verify this conclusion. The update formula of the gating mechanism in the SConvLSTM unit is as follows:

$$\begin{aligned} i_t &= \sigma\left(U_i \circledast h_{t-1} + b_i\right), \\ f_t &= \sigma\left(U_f \circledast h_{t-1} + b_f\right), \\ o_t &= \sigma\left(U_o \circledast h_{t-1} + b_o\right). \end{aligned} \tag{5}$$

The SConvLSTM + unit removes the bias $b$ based on the SConvLSTM unit. The amount of network parameters is further reduced. The update formula of the gating mechanism in the SConvLSTM + unit is as follows:

$$\begin{aligned} i_t &= \sigma\left(U_i \circledast h_{t-1}\right), \\ f_t &= \sigma\left(U_f \circledast h_{t-1}\right), \\ o_t &= \sigma\left(U_o \circledast h_{t-1}\right). \end{aligned} \tag{6}$$

*4.2. Sparse Convolutional GRU (SConvGRU) Network.* As shown in Figure 5(a), the ConvGRU unit is proposed by Ballas et al. [10] in 2016 to solve the problem of video representation. It is also composed of a main line part and gating units. Compared with ConvLSTM, it has fewer gating mechanisms and parameters, and it reduces the transmission of cell state in subsequent networks. The input of the ConvGRU unit includes the input $x_t$ of the current time interval and the hidden state $h_{t-1}$ of the previous time interval, and the output is the hidden state $h_t$ of the current time interval. The main line part is about the update of the hidden state. The updated formula is as follows:

$$\begin{aligned} \widetilde{h_t} &= g\left(W_h \circledast x_t + U_h \circledast \left(r_t \odot h_{t-1}\right) + b_h\right), \\ h_t &= \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \widetilde{h_t}. \end{aligned} \tag{7}$$

where $z_t, r_t$ represent the updating gate and the reset gate, respectively. $h_t$ is the updating the hidden state. $\widetilde{h_t}$ represents the input update. $W$ and $b$ are the learning parameters.

The gating units are composed of an update gate and a reset gate. The reset gate determines the importance of the hidden state $h_{t-1}$ of the previous time interval to the input update $\widetilde{h_t}$, and the update gate determines the influence of the current input state $x_t$ on $h_t$. Like ConvLSTM, the input of the gating mechanism of the ConvGRU unit includes the input state $x_t$ of the current time interval, the hidden state $h_{t-1}$ of the previous time interval, and the bias \$b\$. The specific update equation is as follows:

$$\begin{aligned} z_t &= \sigma\left(W_z \circledast x_t + U_z \circledast h_{t-1} + b_z\right), \\ r_t &= \sigma\left(W_r \circledast x_t + U_r \circledast h_{t-1} + b_r\right), \end{aligned} \tag{8}$$

where $W$, U, and $b$ are the learning parameters.

The SConvGRU unit is shown in Figure 5(b). Like the SConvLSTM unit, we introduce a sparse gating mechanism to reduce the parameters of the reset gate and the update gate. The main line part of SConvGRU is consistent with ConvGRU, and the input of the gating mechanism is composed of the hidden state $h_{t-1}$ of the previous time interval and bias $b$. The current input state $x_t$ is brought into the subsequent spatiotemporal sequence propagation by the input update formula. Compared with ConvGRU, SConvGRU can help to effectively reduce the number of parameters and the training time of the network while keeping a similar prediction accuracy. The updated equation of the gating mechanism in the SConvGRU unit is as follows:

$$\begin{aligned} z_t &= \sigma\left(U_z \circledast h_{t-1} + b_z\right), \\ r_t &= \sigma\left(U_r \circledast h_{t-1} + b_r\right), \end{aligned} \tag{9}$$

where U and $b$ are the learning parameters.

The SConvGRU + unit is based on the SConvGRU unit, and we further remove the bias $b$ to reduce the amount of network parameters. The updated formula of the gating mechanism in the SConvGRU + unit is as follows:

$$\begin{aligned} z_t &= \sigma\left(U_z \circledast h_{t-1}\right), \\ r_t &= \sigma\left(U_r \circledast h_{t-1}\right), \end{aligned} \tag{10}$$

where U is the learning parameter.

*4.3. The Overall Framework of Traffic Flow Prediction.* In this paper, we use the four algorithms: SConvLSTM, SConvLSTM+, SConvGRU, and SConvGRU + to forecast traffic flow. The overall framework of traffic flow prediction is shown in Figure 6, in which the input of the model is the historical traffic flow matrix. Firstly, we use a multilayer convolutional neural network to extract the spatial dependence of the traffic data. Then, a multilayer network built by the above-mentioned units (SConvLSTM unit, SConvLSTM + unit, SConvGRU unit, and SConvGRU + unit) is used to capture the time dependence while further extracting spatial features. In addition, a multilayer deconvolutional neural network is employed to obtain a prediction matrix with the same dimension as the input.

## 5. Experiment

In this section, we evaluate our proposed method based on three real-world traffic flow datasets (TaxiBJ, BikeNYC, and TaxiNYC). Then, we describe our experimental process in detail from the aspects of datasets description, model comparison, evaluation metric, hyperparameter setting, and experimental analysis.

*5.1. Datasets Description.* As shown in Table 1, The TaxiBJ dataset is collected from Beijing taxi GPS data from 07/01/2013 to 10/30/2013. Before the experiment, we, firstly, divide

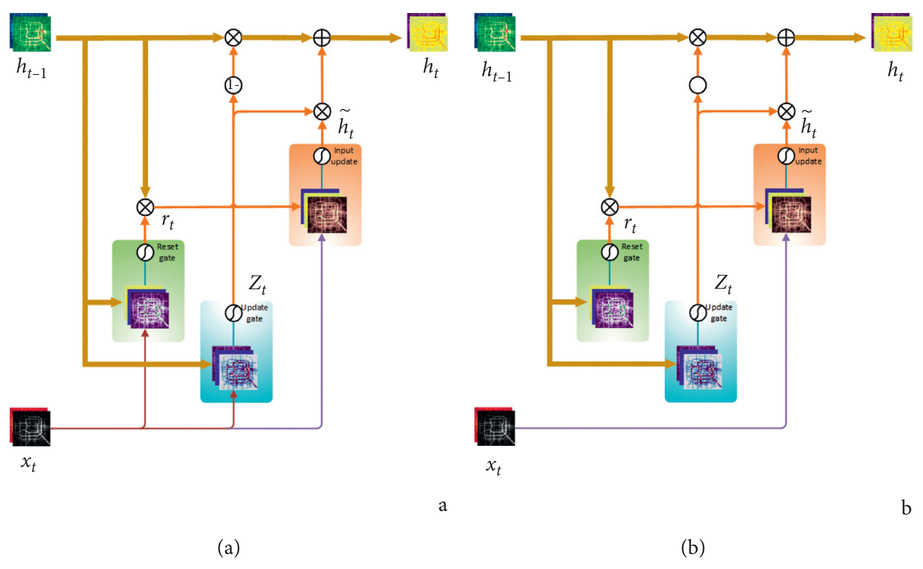


FIGURE 5: (a) ConvGRU unit. (b) SConvGRU unit.

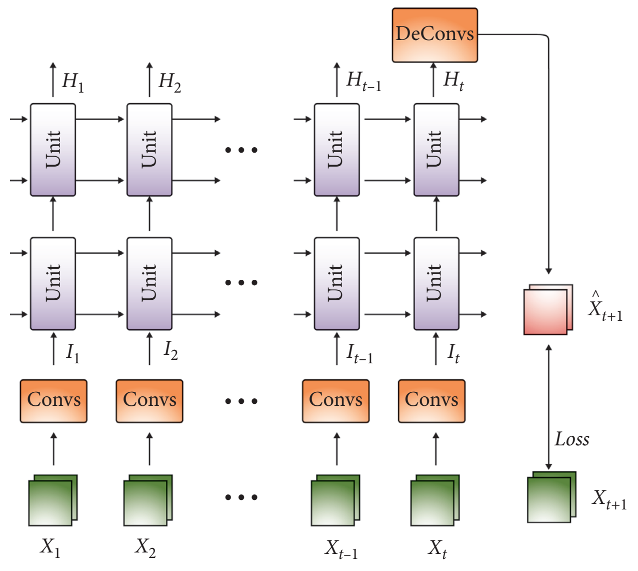


FIGURE 6: Traffic flow prediction model. $\{X_1, X_2, \ldots, X_t\}$ represents the historical traffic flow data. $\{I_1, I_2, \ldots, I_t\}$ denotes the output of the multilayer convolutional neural network. $\{H_1, H_2, \ldots, H_t\}$ represents the output of the multilayer network composed of the above-mentioned units. Convs denotes the multilayer convolutional neural network, and Deconvs denotes the multilayer deconvolutional neural network.

the city into 32 × 32 areas, set the time interval to 30 minutes, and count the traffic flow matrix within each 30 minutes. Furthermore, the data are divided into three subsets: the training sets is the data from 07/01/2013 to 10/20/2013, the validation set is the data from 10/21/2013 to 10/25/2013, and the test set is the data from 10/26/2013 to 10/30/2013.

The TaxiNYC dataset comes from the New York Taxi System from 01/01/2015 to 03/01/2015. For this dataset, we divide the city into 10 × 20 areas and set the time interval to 0.5 hour. In the data, we set the data from 01/01/2015 to 02/09/2015 and from 02/10/2015 to 02/19/2015 as the training set and validation set in the experiment, respectively. The rest of the data is used as the test set in the experiment.

The BikeNYC dataset comes from the New York Citi bike system from 01/01/2016 to 06/30/2016. For this dataset, we, firstly, divide the city into 16 × 16 areas and set the time interval to 1 hour. We set the training set, validation set, and test set to the data from 01/01/2016 to 6/10/2016, from 06/11/2016 to 06/20/2016, and from 06/21/2016 to 06/30/2016, respectively.

### 5.2. Baseline.

*5.2. Baseline.* In this paper, we employ the subsequent model to compare with our proposed model.

(i) HA [13]: HA is a classic time series forecasting method, which predicts the traffic flow using historical average

(ii) ARMA [14]: ARMA is a time series forecasting method for stationary series data, which is based on the AR model and MA model to predict future traffic flow

(iii) VAR [6]: VAR uses the form of simultaneous equations to obtain the linear relationship between different traffic flows

(iv) **ConvLSTM** [8]: ConvLSTM is a deep learning approach for spatiotemporal sequence prediction, which introduces the convolutional structure into the fully connected LSTM

(v) ConvGRU [10]: ConvGRU employs convolutional operation to capture spatial features while preserving the structure of the gated recurrent unit

*5.3. Evaluation Metric.* In the experiment, we predict the traffic flow of the next time interval based on the previous 10-steps of historical traffic data. We use the root mean square error (RMSE) as an evaluation metric to evaluate the

Table 1: The details of the datasets.

| Dataset | TaxiBJ | TaxiNYC | BikeNYC |
|---|---|---|---|
| Start time | 7/1/2013 | 1/1/2015 | 1/1/2016 |
| End time | 10/30/2013 | 3/1/2015 | 6/30/2016 |
| Training set | 7/1/2013–10/20/2013 | 1/1/2015–2/9/2015 | 1/1/2016–6/10/2016 |
| Validation set | 10/21/2013–10/25/2013 | 2/10/2015–2/19/2015 | 6/11/2016–6/20/2016 |
| Test set | 10/26/2013–10/30/2013 | 2/20/2015–3/1/2015 | 6/21/2016–6/30/2016 |
| Time interval | 0.5 | 0.5 | 1 (hour) |
| Grid map size | (32, 32) | (10, 20) | (16, 16) |

performance of our proposed model. The formula is as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i}\sum_{j}\left(x_{ij} - \widehat{x_{ij}}\right)^2}, \tag{11}$$

where $x_{ij}$ denotes the ground truth of the region $g(i, j)$, $\widehat{x_{ij}}$ represents the predicted value of the region $g(i, j)$, and $N$ represents the number of prediction areas.

### 5.4. Experiment on TaxiBJ Dataset

*5.4.1. Hyperparameter Setting.* For the TaxiBJ dataset, a Min-Max normalization is used to scale the data to $[0, 1]$ at first. The overall framework of the model is shown in Section 4.3, and the detailed settings are as follows: a two-layer convolutional neural network is employed to extract the spatial features of the data. After each convolutional operation, we use batch normalization and the rectified linear unit (ReLU). Furthermore, the number of layers of the units (SConvLSTM unit, SConvLSTM + unit, SConvGRU unit, and SConvGRU + unit) is set to 1. In addition, a two-layer deconvolutional neural network is used to obtain the predicted value of the same dimension as the real value. The detailed description of each module is shown in Table 2.

*5.4.2. Convergence Analysis.* Figures 7(a) and 7(b) are the loss curves of the ConvGRU model, SConvGRU model, and SConvGRU + model on the training set and validation set. From the figure, we can see that with the increase of training times, the RMSE on the training set and the validation set gradually decreases, and finally converges. Furthermore, we can see that the SConvGRU + model converges to a lower RMSE on the validation set, compared to the ConvGRU model and SConvGRU model. In addition, on the training set, the three models maintain a consistent convergence trend. In summary, the convergence speed and performance of our proposed methods do not reduce under the condition of compressing the original models.

Figures 7(c) and 7(d) are the loss curves of the ConvLSTM model, SConvLSTM model, and SConvLSTM + model on the training set and validation set. We can see from the figure that the three models basically maintain consistent convergence performance on the training set and validation set. Similarly, the model achieves convergence on the validation set. In a word, the convergence performance of our proposed models does not reduce.

*5.4.3. The Comparative Results of Different Models.* Table 3 describes the RMSE comparison between the traditional model and our proposed model on the test set. First of all, we can see that the deep learning method is better than the traditional predictive model, which shows that the deep learning method can better capture the nonlinear characteristics of the data. Compared with the ConvGRU model, the RMSE of the SConvGRU + model increases by 1.8%. This result shows that the parameters of the model are effectively reduced under the condition of losing limited prediction accuracy. In addition, compared with ConvLSTM, SConvLSTM + improves the prediction accuracy of the model by 4.5% while reducing the number of model parameters. In general, SConvLSTM achieves the best performance on the TaxiBJ dataset. This result shows the effectiveness of the sparse convolutional recurrent network.

*5.4.4. The Rate of Model Compression.* Table 4 shows the comparison of the rate of model's compression. We use ConvGRU and ConvLSTM as the baselines to calculate the rate of parameter reduction of our proposed models. Compared with ConvGRU, the rates of parameter reduction of SConvGRU and SConvGRU + are 13.3% and 13.5%, respectively. Compared with ConvLSTM, the rates of the parameter reduction of SConvLSTM and SConvLSTM + are 15% and 15.1%, respectively. When the same hyperparameters in Table 2 are set, the number of the parameters of ConvLSTM, SConvLSTM, and SConvLSTM + are more than ConvGRU, SConvGRU, and SConvGRU+, respectively. The reason for this result is that ConvLSTM, SConvLSTM, and SConvLSTM + have more gating mechanism than ConvGRU, SConvGRU, and SConvGRU+.

### 5.5. Experiment on TaxiNYC Dataset

*5.5.1. Hyperparameter Setting.* For the TaxiNYC dataset, similarly, we first use the Min-Max normalization approach to scale the original data to $[0, 1]$. The framework of the model is shown in Section 4.3. Different to the setting on the TaxiBJ dataset, the number of layers of the units (SConvLSTM unit, SConvLSTM + unit, SConvGRU unit, and SConvGRU + unit) are set to 2, and the hidden channels of ConvGRU unit, SConvGRU unit, and SConvGRU + unit are set to 32 (first layer) and 64 (second layer). Furthermore, the hidden channels of ConvLSTM unit, SConvLSTM unit, and SConvLSTM + unit are set to 32 (first layer) and 32 (second layer). Then, the detailed settings are shown in Table 5. In addition, other hyperparameter

Table 2: The details of module on TaxiBJ dataset.

| Hyperparameter | ConvGRU | SConvGRU | SConvGRU+ | ConvLSTM | SConvLSTM | SConvLSTM+ |
|---|---|---|---|---|---|---|
| CNN layers | 2 | 2 | 2 | 2 | 2 | 2 |
| Number of filters in CNN | 8 (first layer), 16 (second layer) | | | | | |
| Kernel size in CNN | (3, 3) | (3, 3) | (3, 3) | (3, 3) | (3, 3) | (3, 3) |
| Stride in CNN | 1 (first layer), 2 (second layer) | | | | | |
| Unit layers | 1 | 1 | 1 | 1 | 1 | 1 |
| Hidden channels of unit | 64 | 64 | 64 | 64 | 64 | 64 |
| Kernel size in unit | (3, 3) | (3, 3) | (3, 3) | (3, 3) | (3, 3) | (3, 3) |
| Input of gating mechanism | $h_{t-1}, x_t, b$ | $h_{t-1}, b$ | $h_{t-1}$ | $h_{t-1}, x_t, b$ | $h_{t-1}, b$ | $h_{t-1}$ |
| DCNN layers | 2 | 2 | 2 | 2 | 2 | 2 |
| Number of filters in DCNN | 8 (first layer), 2 (second layer) | | | | | |
| Kernel size in DCNN | (3, 3) | (3, 3) | (3, 3) | (3, 3) | (3, 3) | (3, 3) |
| Stride in DCNN | 2 (first layer), 1 (second layer) | | | | | |
| Batch size | 16 | 16 | 16 | 16 | 16 | 16 |
| Timestep | 10 | 10 | 10 | 10 | 10 | 10 |
| Epoch | 300 | 300 | 300 | 300 | 300 | 300 |
| Optimizer | Adam [36] | Adam | Adam | Adam | Adam | Adam |
| Learning rate | 0.0002 | 0.0002 | 0.0002 | 0.0002 | 0.0002 | 0.0002 |
| Strategy | Early-stopping | Early-stopping | Early-stopping | Early-stopping | Early-stopping | Early-stopping |

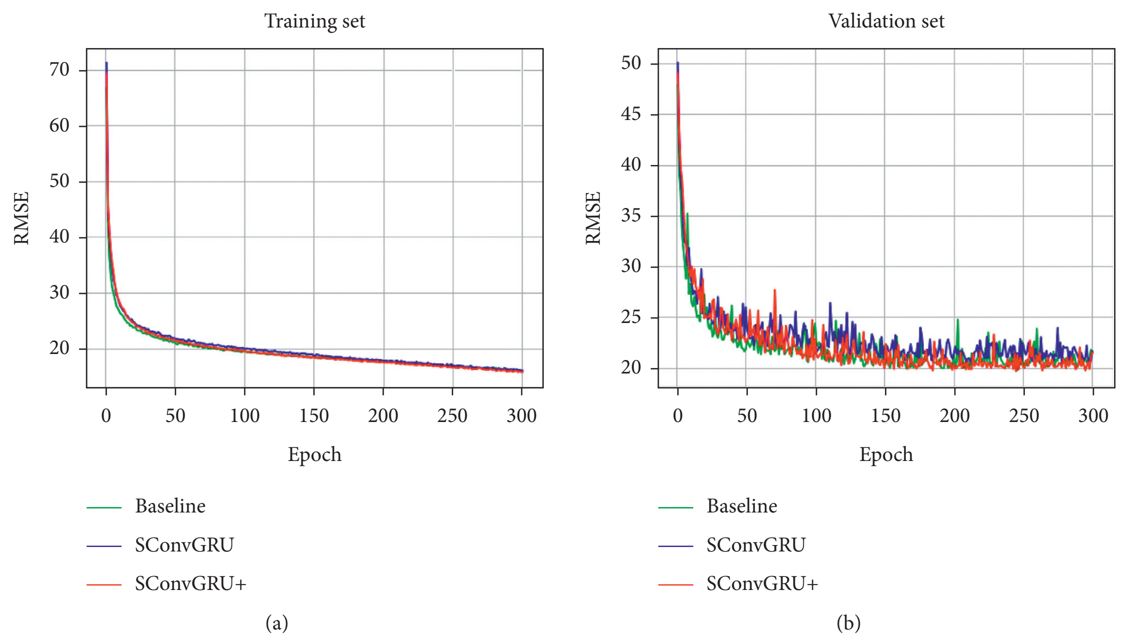
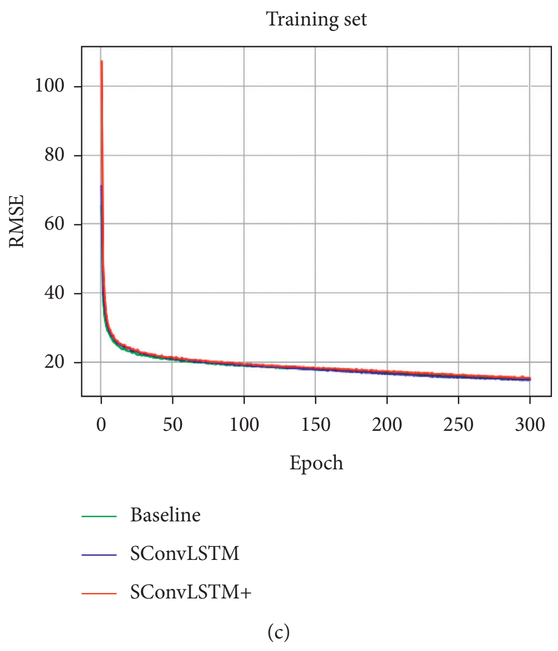


Figure 7: Continued.



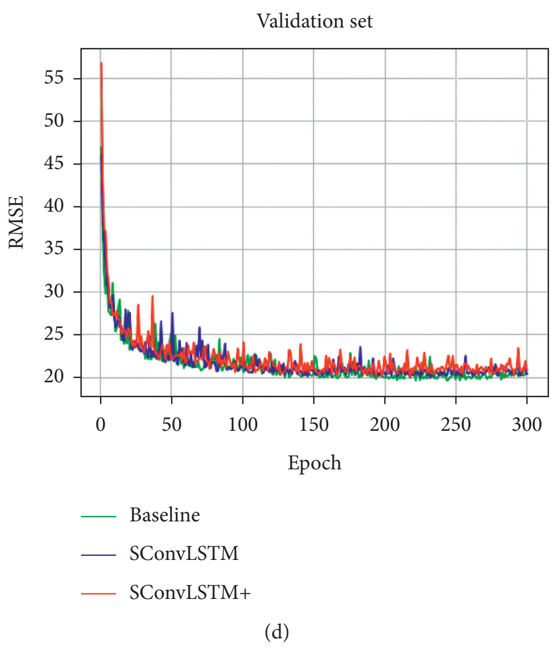


Figure 7: The loss curve of the model on the TaxiBJ dataset. (a) ConvGRU, SConvGRU, and SConvGRU + on the training set. (b) ConvGRU, SConvGRU, and SConvGRU + on the validation set. (c) ConvLSTM, SConvLSTM, and SConvLSTM + on the training set. (d) ConvLSTM, SConvLSTM, and SConvLSTM + on the validation set.

Table 3: The comparative results of different models.

| Model | RMSE |
|---|---|
| HA | 97.348 |
| ARMA | 21.324 |
| VAR | 22.945 |
| ConvGRU | 19.290 |
| SConvGRU | 19.484 |
| SConvGRU+ | 19.632 |
| ConvLSTM | 19.637 |
| SConvLSTM | 19.270 |
| SConvLSTM+ | **18.751** |

Bold represents the best results.

Table 4: The rate of model compression.

| Model | Parameter | Rate of parameter reduction |
|---|---|---|
| ConvGRU | 138432 | 0 |
| SConvGRU | 120000 | 13.3% |
| SConvGRU+ | 119808 | 13.5% |
| ConvLSTM | 184576 | 0 |
| SConvLSTM | 156928 | 15% |
| SConvLSTM+ | 156672 | 15.1% |

settings are shown in Table 2, which are the same as those of the TaxiBJ dataset.

*5.5.2. Convergence Analysis.* Figures 8(a) and 8(b) are the loss curves of the ConvGRU model, SConvGRU model, and SConvGRU + model on the training set and validation set. From the figure, it can be seen that the three models maintain a consistent convergence curve. Furthermore, RMSE gradually decreases with the increase of epoch on the training set and validation set and finally converges.

Figures 8(c) and 8(d) are the loss curves of the ConvLSTM model, SConvLSTM model, and SConvLSTM + model. Firstly, the model achieves convergence on the validation set. Then, their convergence on the training set and validation set is basically the same, and the baseline model on the validation set converges to a lower value.

*5.5.3. The Comparative Results of Different Models.* From Table 6, in the traditional model, VAR achieved a good result of 14.306. Compared with ConvGRU, the RMSE of SConvGRU+ is increased by 0.07. Similarly, compared with ConvLSTM, the RMSE of SConvLSTM+ is increased by 0.029. This result shows that the parameters of the model are effectively reduced under the condition of losing limited prediction accuracy. On the whole, the prediction results of the ConvLSTM model, SConvLSTM model, and SConvLSTM + model are better than the ConvGRU model, SConvGRU model, and SConvGRU + model. This result shows that ConvLSTM and sparse ConvLSTM can better capture the nonlinear structures on this dataset.

*5.5.4. The Rate of Model Compression.* The rate of compression on different models are shown in Table 7 on the TaxiNYC dataset. Because of the different number of hidden channels, the number of parameters of ConvLSTM-based models is less than that of ConvGRU-based models. Compared with the ConvGRU model, the compression rates of the SConvGRU model and the SConvGRU + model are 22.2% and 22.3%, respectively. Compared with the ConvLSTM model, the compression rates of the SConvLSTM model and the SConvLSTM + model are 32.1% and 32.3%, respectively.

Table 5: The details of the module on the TaxiNYC dataset.

| Hyperparameter | ConvGRU | SConvGRU | SConvGRU+ | ConvLSTM | SConvLSTM | SConvLSTM+ |
|---|---|---|---|---|---|---|
| Kernel size in CNN | (3, 5) | (3, 5) | (3, 5) | (3, 5) | (3, 5) | (3, 5) |
| Stride in CNN | 1 (first layer), 1 (second layer) | | | | | |
| Unit layers | 2 | 2 | 2 | 2 | 2 | 2 |
| Hidden channels of unit | 32 (first layer), 64 (second layer) | | | 32 (first layer), 32 (second layer) | | |
| Kernel size in DCNN | (3, 5) | (3, 5) | (3, 5) | (3, 5) | (3, 5) | (3, 5) |
| Stride in DCNN | 1 (first layer), 1 (second layer) | | | | | |

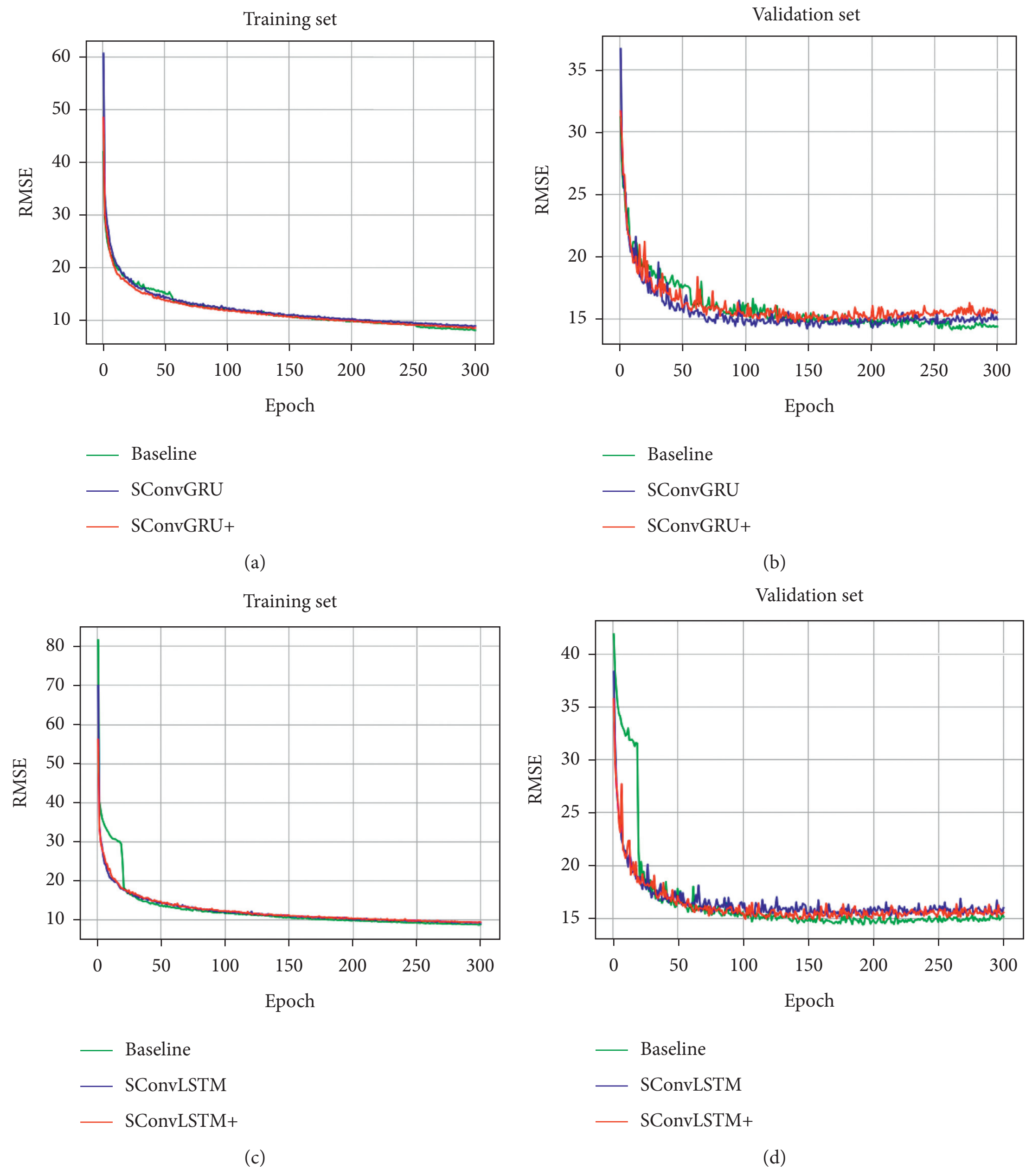


Figure 8: The loss curve of the model on the TaxiNYC dataset. (a): ConvGRU, SConvGRU, and SConvGRU + on the training set. (b) ConvGRU, SConvGRU, and SConvGRU + on the validation set. (c) ConvLSTM, SConvLSTM, and SConvLSTM + on the training set. (d) ConvLSTM, SConvLSTM, and SConvLSTM + on the validation set.

Table 6: The comparative results of different models.

| Model | RMSE |
|---|---|
| HA | 39.764 |
| ARMA | 15.658 |
| VAR | 14.306 |
| ConvGRU | **12.851** |
| SConvGRU | 13.059 |
| SConvGRU+ | 12.921 |
| ConvLSTM | **12.373** |
| SConvLSTM | 12.554 |
| SConvLSTM+ | 12.402 |

Bold represents the best results.

Table 7: The rate of model compression.

| Model | Parameter | Rate of parameter reduction |
|---|---|---|
| ConvGRU | 207648 | 0 |
| SConvGRU | 161568 | 22.2% |
| SConvGRU+ | 161280 | 22.3% |
| ConvLSTM | 129280 | 0 |
| SConvLSTM | 87808 | 32.1% |
| SConvLSTM+ | 87552 | 32.3% |

### 5.6. Experiment on BikeNYC Dataset

*5.6.1. Hyperparameter Setting.* Similarly, the overall framework of the model is shown in Section 4.3, which consists of a two-layer convolutional neural network, a two-layer unit, and a two-layer deconvolutional neural network. In addition, the specific details of the framework are shown in Table 8. Other hyperparameter settings are shown in Table 2, which are the same as those in the TaxiBJ dataset.

*5.6.2. Convergence Analysis.* Figures 9(a) and 9(b) are the loss curves of the ConvGRU model, SConvGRU model, and SConvGRU + model on the training set and validation set. From Figure 9(a), we can see that the RMSE of the model gradually decreases with the increase of epoch. Furthermore, we can find that the baseline (ConvGRU) is reduced to a lower RMSE compared to SConvGRU and SConvGRU+ in the training process. However, ConvGRU has worse performance in the validation set compared with our proposed models. From Figure 9(b), we can see that the RMSE of the model decreases first and then increases with the increase of the epoch. The reason for this result may be that the model is overfitting. In addition, the SConvGRU model and the SConvGRU + model achieve better performance on the validation set, which proves that the prediction accuracy is not reduced under the condition of reducing the number of parameters.

Figures 9(c) and 9(d) are the loss curves of the ConvLSTM model, SConvLSTM model, and SConvLSTM + model. Like Figures 9(a) and 9(b), we can see that a lower RMSE on the training set can be gained by ConvLSTM compared to SConvLSTM and SConvLSTM+, however, ConvLSTM on the validation set shows worse performance than SConvLSTM and SConvLSTM+. The reason for this result may be that when training epoches exceed a certain number, the model is overfitting. On the validation set, the SConvLSTM model and the SConvLSTM + model achieve a lower RMSE.

*5.6.3. The Comparative Results of Different Models.* In the BikeNYC dataset, it can be seen from Table 9 that SConvLSTM achieves the best performance. Compared with the ConvGRU model, the SConvGRU model and the SConvGRU + model achieve a lower RMSE. Similarly, compared with the ConvLSTM model, the SConvLSTM model and the SConvLSTM + model also achieve better performance. This result proves that under the condition of reducing the parameters of the model, the prediction effect of the model is improved at the same time. The reason for this result maybe that part of the parameters of the gating mechanism in ConvLSTM and ConvGRU are redundant. Furthermore, on the whole, compared with the ConvGRU model, the SConvGRU model, and the SConvGRU + model, the ConvLSTM model, the SConvLSTM model, and SConvLSTM + model achieve better RMSE, which shows that the ConvLSTM model and the sparse long short-term memory network have a stronger ability to capture the spatiotemporal information.

*5.6.4. The Rate of Model Compression.* Table 10 shows the rate of compression on different models on the BikeNYC dataset. The ConvGRU model and ConvLSTM model are used as the benchmark to measure the rate of compression of our proposed model. Compared with the ConvGRU model, the compression rates of the SConvGRU model and the SConvGRU + model are 25.6% and 25.7%, respectively. Compared to the ConvLSTM model, the compression rates of the SConvLSTM model and the SConvLSTM + model are 28.8% and 28.9%, respectively.

Table 8: The details of the module on the TaxiNYC dataset.

| Parameter | ConvGRU | SConvGRU | SConvGRU+ | ConvLSTM | SConvLSTM | SConvLSTM+ |
|---|---|---|---|---|---|---|
| Stride in CNN | 1 (first layer), 1 (second layer) | | | | | |
| Unit layers | 2 | 2 | 2 | 2 | 2 | 2 |
| Stride in DCNN | 1 (first layer), 1 (second layer) | | | | | |

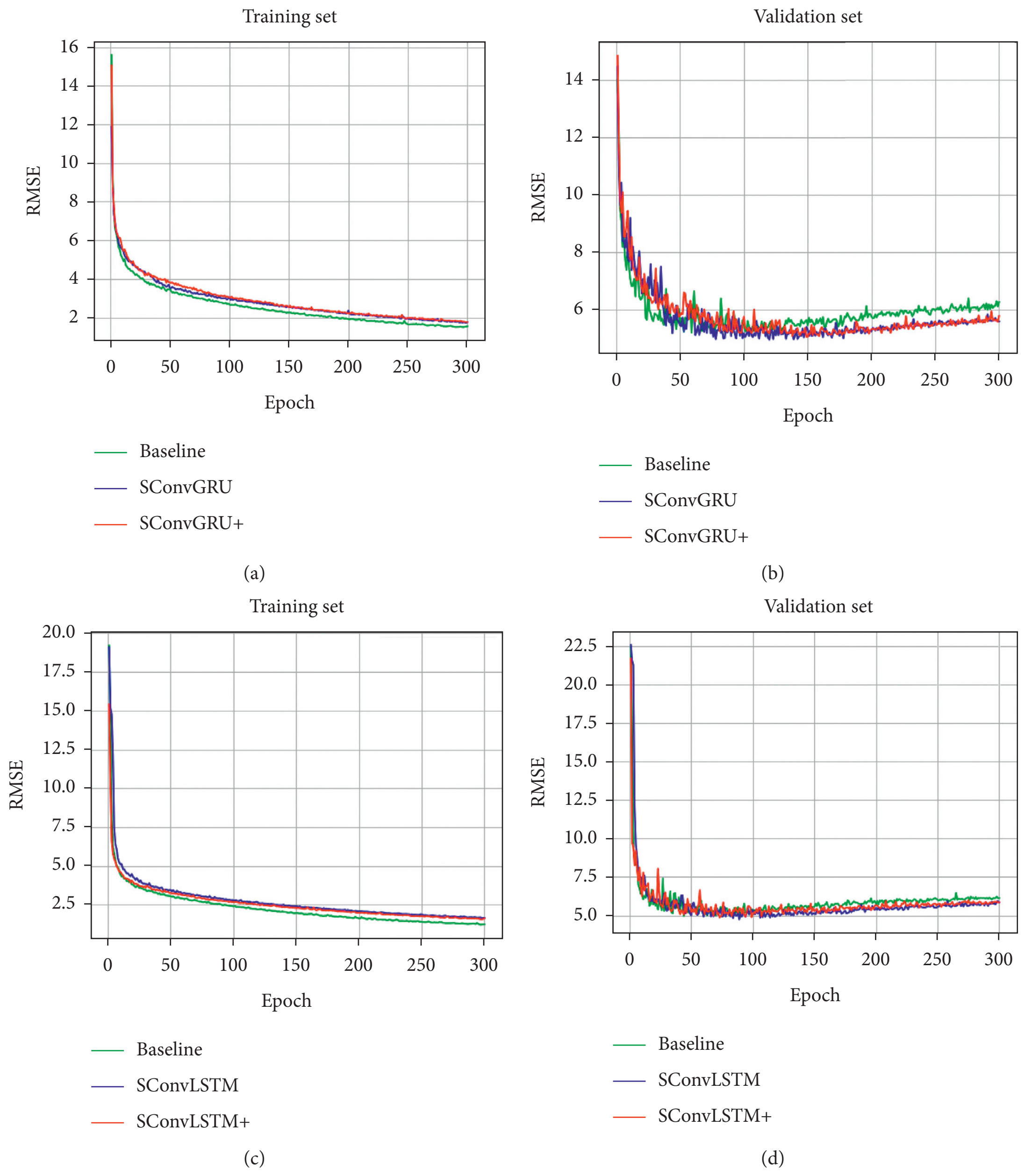


Figure 9: The loss curve of the model on the training set and validation set on the BikeNYC dataset. (a, b) ConvGRU, SConvGRU, and SConvGRU+. (c, d) ConvLSTM, SConvLSTM, and SConvLSTM+.

Table 9: The comparative results of different models.

| Model | RMSE |
|---|---|
| HA | 11.968 |
| ARMA | 15.463 |
| VAR | 6.478 |
| ConvGRU | 5.547 |
| SConvGRU | 5.084 |
| SConvGRU+ | 5.259 |
| ConvLSTM | 4.994 |
| SConvLSTM | **4.852** |
| SConvLSTM+ | 4.973 |

Bold represents the best results.

Table 10: The rate of model compression.

| Model | Parameter | Rate of parameter reduction |
|---|---|---|
| ConvGRU | 359808 | 0 |
| SConvGRU | 267648 | 25.6% |
| SConvGRU+ | 267264 | 25.7% |
| ConvLSTM | 479744 | 0 |
| SConvLSTM | 341504 | 28.8% |
| SConvLSTM+ | 340992 | 28.9% |

## 6. Conclusion and Future Work

In this paper, we study how to compress the convolutional recurrent networks while keeping the competitive results with traditional algorithms. To solve this problem, we propose a sparse convolutional recurrent network framework, in which a sparse gating mechanism is developed. For ConvGRU, we develop the SConvGRU unit and SConvGRU + unit. For ConvLSTM, we develop the SConvLSTM unit and SConvLSTM + unit. Based on three real traffic flow datasets, the experimental results prove that our proposed methods are able to effectively reduce the number of model parameters while keeping the prediction accuracy.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors have no conflicts of interest.

## Acknowledgments

This research was supported by the National Natural Science Foundation of China (nos. 62062033 and 62067002), the Natural Science Foundation of Jiangxi Province (Nos. 20212BAB202008 and 20192ACBL21006), Key Research and Develop Project of Jiangxi Province (no. 20203BBE53034), and the Education Department Project of Jiangxi Province (GJJ200604).

## References

[1] Y. Le Cun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Esteva, K. Chou, S. Yeung et al., "Deep learning-enabled medical computer vision," *NPJ Digital Medicine*, vol. 4, pp. 1–9, 2021.

[3] J. Xie, Y. Zheng, R. Du et al., "Deep learning-based computer vision for surveillance in its: evaluation of state-of-the-art methods," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3027–3042, 2021.

[4] M. Giménez, J. Palanca, and V. Botti, "Semantic-based padding in convolutional neural networks for improving the performance in natural language processing. a case of study in sentiment analysis," *Neurocomputing*, vol. 378, pp. 315–323, 2020.

[5] H. Zheng, F. Lin, X. Feng, and Y. Chen, "A hybrid deep learning model with attention-based conv-lstm networks for short-term traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, 2020.

[6] M. Lv, Z. Hong, L. Chen, T. Chen, T. Zhu, and S. Ji, "Temporal multi-graph convolutional network for traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, 2020.

[7] L. Liu, J. Zhen, G. Li et al., "Dynamic spatial-temporal representation learning for traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 11, 2020.

[8] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Convolutional lstm network: a machine learning approach for precipitation nowcasting," *Advances in Neural Information Processing Systems*, vol. 1, pp. 802–810, 2015.

[9] A. Graves, "Generating sequences with recurrent neural networks," 2013, https://arxiv.org/abs/1308.0850.

[10] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," in *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016*, San Juan, Puerto Rico, May 2016.

[11] A. Zonoozi, J. J. Kim, X. L. Li, and G. Cong, "Periodic-crn: a convolutional recurrent model for crowd density prediction with recurring periodic patterns," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3732–3738, Stockholm, Sweden, July 2018.

[12] D. Zang, J. Ling, Z. Wei, K. Tang, and J. Cheng, "Long-term traffic speed prediction based on multiscale spatio-temporal feature learning network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, pp. 3700–3709, 2018.

[13] J. Liu and W. Guan, "A summary of traffic flow forecasting methods," *Journal of Highway and Transportation Research and Development*, vol. 21, pp. 82–85, 2004.

[14] J. Klepsch, C. Klüppelberg, and T. Wei, "Prediction of functional arma processes with an application to traffic data," *Econometrics and Statistics*, vol. 1, pp. 128–149, 2017.

[15] C. Chen, J. Hu, Q. Meng, and Y. Zhang, "Short-time traffic flow prediction with arima-garch model," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 607–612, Baden-Baden, Germany, June 2011.

[16] S. R. Chandra and H. Al-Deek, "Predictions of freeway traffic speeds and volumes using vector autoregressive models," *Journal of Intelligent Transportation Systems*, vol. 13, no. 2, pp. 53–72, 2009.

[17] 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, Article ID 664672, 2003.

[18] M. Cools, E. Moons, and G. Wets, "Investigating the variability in daily traffic counts through use of ARIMAX and SARIMAX models," *Transportation Research Record: Journal*

*of the Transportation Research Board*, vol. 2136, no. 1, pp. 57–66, 2009.

[19] Y. Kamarianakis and P. Prastacos, "Forecasting traffic flow conditions in an urban network: comparison of multivariate and univariate approaches," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1857, no. 1, pp. 74–84, 2003.

[20] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in *Proceedings of the 31st Youth Academic Annual Conference of Chinese Association of Automation*, pp. 324–328, Wuhan, China, November 2016.

[21] X. Dai, R. Fu, Y. Lin, L. Li, and F. Y. Wang, "Deeptrend: a deep hierarchical neural network for traffic flow prediction," 2017, https://arxiv.org/abs/1707.03213.

[22] D. Kang, Y. Lv, and Y Y. Chen, "Short-term traffic flow prediction with lstm recurrent neural network," in *Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems*, pp. 1–6, Yokohama, Japan, October 2017.

[23] X. Luo, D. Li, Y. Yang, and S. Zhang, "Spatiotemporal traffic flow prediction with knn and lstm," *Journal of Advanced Transportation*, vol. 2019, Article ID 4145353, 10 pages, 2019.

[24] L. Zhang, Q. Liu, W. Yang, N. Wei, and D. Dong, "An improved k-nearest neighbor model for short-term traffic flow prediction," *Procedia - Social and Behavioral Sciences*, vol. 96, pp. 653–662, 2013.

[25] D. Wang, Y. Yang, and S. Ning, "Deepstcl: a deep spatio-temporal convlstm for travel demand prediction," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–8, Rio de Janeiro, Brazil, July 2018.

[26] X. Chen, X. Xie, and D. Teng, "Short-term traffic flow prediction based on convlstm model," in *Proceedings of the 5th IEEE Information Technology and Mechatronics Engineering Conference*, Article ID 846850, Chongqing, China, June 2020.

[27] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[28] P. Li, M. Sun, and M. Pang, "Prediction of taxi demand based on convlstm neural network," *Neural Information Processing*, in *Proceedings of the International Conference on Neural Information Processing*, pp. 15–25, Sanur Bali, Indonesia, December 2018.

[29] C. Louizos, M. Welling, and D. P. Kingma, "Learning Sparse Neural Networks through L_0 Regularization," in *Proceedings of the International Conference on Learning Representations*, Vancouver, BC, Canada, May 2018.

[30] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Article ID 806814, Boston, MA, USA, June 2015.

[31] A. Parashar, M. Rhu, A. Mukkara et al., "Scnn," *ACM SIGARCH - Computer Architecture News*, vol. 45, no. 2, pp. 27–40, 2017.

[32] T. Dettmers and L. Zettlemoyer, "Sparse Networks from Scratch: Faster Training without Losing Performance," 2019, https://arxiv.org/abs/1907.04840.

[33] S. Alford, R. Robinett, L. Milechin, and J. Kepner, "Pruned and structurally sparse neural networks," in *Proceedings of the IEEE MIT Undergraduate Research Technology Conference*, pp. 1–4, Cambridge, MA, USA, October 2018.

[34] J. H. Luo, J. Wu, and W. Lin, "Thinet: a filter level pruning method for deep neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5058–5066, Venice, Italy, October 2017.

[35] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "Dnn-based prediction model for spatio-temporal data," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 1–4, Burlingame, CA, USA, November 2016.

[36] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May. 2015.