

Deep Learning with Long Short-Term Memory for Time Series Prediction

Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, and Honggang Zhang

Y. Hua, Z. Zhao, R. Li and H. Zhang are with Zhejiang University, Hangzhou 310027, China, (email: {21631087, zhaozf, lironpeng, honggangzhang}@zju.edu.cn). X. Chen is with VTT Technical Research Centre of Finland, Oulu FI-90571, Finland (email: xianfu.chen@vtt.fi). Z. Liu is with China Mobile Research Institute, Beijing 100053, China (email: liuzhiming@chinamobile.com).

Abstract

Time series prediction can be generalized as a process that extracts useful information from historical records and then determines future values. Learning long-range dependencies that are embedded in time series is often an obstacle for most algorithms, whereas Long Short-Term Memory (LSTM) solutions, as a specific kind of scheme in deep learning, promise to effectively overcome the problem. In this article, we first give a brief introduction to the structure and forward propagation mechanism of the LSTM model. Then, aiming at reducing the considerable computing cost of LSTM, we put forward the Random Connectivity LSTM (RCLSTM) model and test it by predicting traffic and user mobility in telecommunication networks. Compared to LSTM, RCLSTM is formed via stochastic connectivity between neurons, which achieves a significant breakthrough in the architecture formation of neural networks. In this way, the RCLSTM model exhibits a certain level of sparsity, which leads to an appealing decrease in the computational complexity and makes the RCLSTM model become more applicable in latency-stringent application scenarios. In the field of telecommunication networks, the prediction of traffic series and mobility traces could directly benefit from this improvement as we further demonstrate that the prediction accuracy of RCLSTM is comparable to that of the conventional LSTM no matter how we change the number of training samples or the length of input sequences.

I INTRODUCTION

The analysis and prediction of time series has always been the key technique in an array of practical problems, including weather forecasting, transportation planning, traffic management, and so on. In the domain of telecommunications, intelligent mechanisms have already been designed to track and analyze a large number of time-dependent events, such as data traffic, user location, channel load, and service requests, to mention a few [1]. On the other hand, with the explosive proliferation of mobile terminals as well as the expansion of mobile Internet, the Internet of Things (IoT) and cloud computing, the mobile communication network has become an indispensable social infrastructure that is bound up with people's lives and various areas of society [1]. However, it remains a challenging issue that how to guarantee the quality of service (QoS) and the quality of experience regardless of the dynamics of network traffic and user movements. One promising solution is to predict the varying pattern of data traffic and the location at which a mobile user will likely demand network service [1]. Accordingly, network operators can reserve some network resources, and react effectively to network changes in near real time [2]. However, since misplaced reservation of network resources and outdated predicted information will not only fail to support the desired QoS but also likely degrade the performance of the overall network, the prediction accuracy and complexity is of vital importance [3].

Basically, the prediction goal of a time series $\{y_1, y_2, \dots\}$ is to estimate the value at time i based on its previous data y_{i-1}, y_{i-2}, \dots . If we denote $\mathbf{x} = \{y_{i-k}, y_{i-k+1}, \dots, y_{i-1}\}$, $i = \{k, \dots, n\}$, the goal aims at finding a function $f(\mathbf{x})$ so that $\hat{y}_i = f(\mathbf{x})$ is as close to the ground truth y_i as possible.

Time series analysis and prediction have been intensively studied for 40 years [4]. In statistical signal processing, the Autoregressive Integrated Moving Average (ARIMA) model has been used to study time-varying processes. However, one limitation of ARIMA is its natural tendency to concentrate on the mean values of the past series data. Therefore, it remains challenging to capture a rapidly changing process [5]. Support Vector Regression (SVR) has been successfully applied for time series prediction, but it also has disadvantages like the lack of structured means to determine some key parameters of the model [5]. In recent years, owing to the flexible structure, deep learning models are increasingly used in time series prediction [6]. Specifically, Recurrent Neural Networks (RNNs), one of deep learning models, establish the reputation to cope with time series by recurrent neural connections. However, for any standard RNN architecture, the influence of a given input on the hidden layers and eventually on the neural network output would either decay or blow up exponentially when cycling around recurrent connections. To tackle this problem, Long Short-Term Memory (LSTM) has been revolutionarily designed by changing the structure of the hidden neurons in traditional RNN [7]. Today, research and applications of LSTM for time series prediction are proliferating. For example, Wang *et al.* [2] used LSTM-based model to predict the next-moment traffic load in a specific geometric area and Alahi *et al.* [3] predicted the motion dynamics in crowded scenes based on LSTM.

Generally, without customized hardware and software acceleration, the LSTM's computing time is proportional to the number of parameters. Given this disappointing characteristic, in this article, we present an approach to decrease the number of involved parameters, and thus put forward a new model that reduces the computational cost. Inspired by the interesting finding that Feed Forward Neural Networks (FFNNs) with sparse neural connections have a similar or even superior performance in many experiments compared to the conventional FFNNs [8], we introduce random connectivity to the conventional LSTM model, thus forming a new architecture with sparse neural connections, called the Random Connectivity Long Short-Term Memory (RCLSTM) model.

Our simulation model is a three-layer stack RCLSTM neural network with a memory cell size of 300 per layer. The simulation data comprise traffic data from GÉANT networks—a pan-European research network [9], and realistic user-trajectory data [10]. The reasons to use the two different datasets are twofold. First, ANN-based models like LSTM are sensitive to the dataset and may yield significant performance changes on different subsets taken from the same database. Therefore, it would be hard to draw conclusions from one single dataset for comparing different algorithms. Second, the prediction of both network traffic and user mobility is of significant importance in the design and optimization of telecommunication networks. For these reasons, we take advantage of the two different datasets. Our simulation results show that when compared to LSTM, the RCLSTM model is highly capable of traffic prediction and user-location forecasting with less than half the neural connections. Particularly, in the traffic prediction task, RCLSTM with even 1% neural connections performs better than ARIMA, SVR, and FFNN, while reducing the computing time by around 30% compared with the conventional LSTM. More interestingly, when the length of input traffic sequences increases from 50 to 500, the prediction error of RCLSTM remains basically unchanged, whereas that of LSTM increases by about 10%.

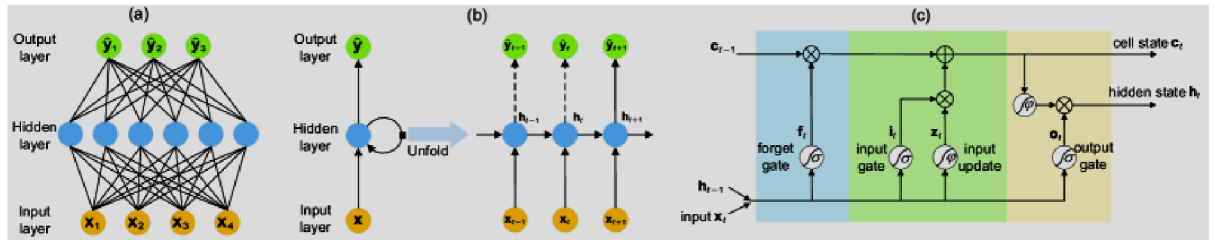


Figure 1: An illustration of FFNN, RNN and LSTM memory block: (a) FFNN; (b) RNN; (c) LSTM memory block.

II AN OVERVIEW OF ARTIFICIAL NEURAL NETWORKS AND LSTM

Artificial Neural Networks (ANNs) are constructed as a class of machine learning models that can eliminate the drawbacks of the traditional learning algorithms with rule-based programming [11]. ANNs can be classi-

fied into two main categories—FFNNs and RNNs. FFNNs usually consist of an input layer, an output layer and hidden layers (if necessary). Each layer is composed of a number of neurons and an activation function. A simple diagram of FFNNs is illustrated in Fig. 1(a). In FFNNs, there is no connection between the neurons within the same layer, and all neurons cannot be connected across layers, which means the information flows in one direction, from the input layer, through the hidden layers (if any), to the output layer. FFNNs are widely used in various fields like data classification, object recognition, and image processing. However, constrained by their internal structure, FFNNs are unsuitable for handling historical dependencies.

RNNs, as another type of ANNs, are similar to FFNNs in the structure of neural layers, but allow the connections between the neurons within the same hidden layer. An illustration of RNNs can be observed in the left-hand side of Fig. 1(b). In addition, the right-hand side of Fig. 1(b) is the expanded form of the RNN model, indicating that RNNs calculate the output of the current moment from the input of the current moment x_t and the hidden state of the previous moment h_{t-1} . Therefore, RNNs allow historical input information to be stored in the network's internal state, and are thereby capable of mapping all of the historical input data to the final output. Theoretically, RNNs are competent to handle such long-range dependencies. However, in practice, RNNs seem unable to accomplish the task. This phenomenon has been explored in depth by Hochreiter and Schmidhuber [7], who explained some pretty fundamental reasons why such learning might be difficult.

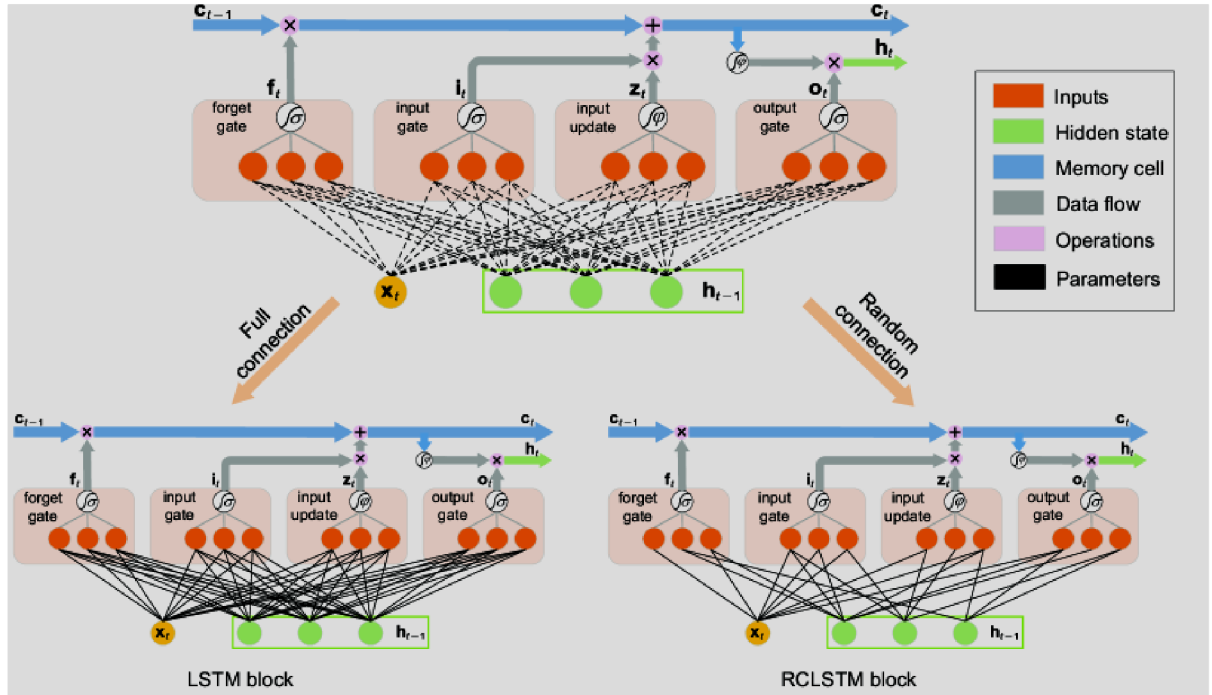


Figure 2: The comparison between LSTM and RCLSTM in the view of generating process of neural connections.

Long Short-Term Memory networks, usually just called “LSTMs”, are a special RNNs that are suitable for learning long-term dependencies [7]. The key part that enhances LSTMs’ capability to model long-term dependencies is a component called memory block [7]. As illustrated in Fig. 1(c), the memory block is a recurrently connected subnet that contains functional modules called the memory cell and gates. The memory cell is in charge of remembering the temporal state of the neural network and the gates formed by multiplicative units are responsible for controlling the pattern of information flow. According to the corresponding practical functionalities, these gates are classified as input gates, output gates and forget gates. Input gates control how much new information flows into the memory cell, while forget gates control how much information of the memory cell still remains in the current memory cell through recurrent connection, and output gates control how much information is used to compute the output activation of the memory block and further flows into the rest of the neural network. Before going through the details of LSTM, some simple yet useful

activation functions need to be reviewed. The sigmoid function $\sigma(x) = \frac{1}{1 + e^{-x}}$ and the tanh function $\varphi(x) = 2\sigma(2x) - 1$ are commonly used as the activation function in ANNs. The domain of both functions is the real number field, but the return value for the sigmoid function ranges from 0 to 1, while the tanh function ranges from -1 to 1. Fig. 1(c) explains in detail how LSTM works. The first step is to decide what kind of information will be removed from the memory cell state, which is implemented by a sigmoid layer (i.e., the forget gate). The next step is to decide what new information will be stored in the memory cell state. This operation can be divided into two steps. First, a sigmoid layer (i.e., the input gate) determines what will be updated, and a tanh layer creates a vector of new candidate values z_t that can be added to the memory cell state, where the subscript t denotes the current moment. Next, these two parts are combined to trigger an update to the memory cell state. To update the old memory cell state c_{t-1} into the new memory cell state c_t , we can first multiply the corresponding elements of c_{t-1} and the output of forget gate layer (i.e. f_t), which is just like the oblivion mechanism in the human brain, and then add $i_t * z_t$, where i_t denotes the output of input gate and $*$ denotes element-wise multiplication. The last step is to decide what to output, which is realized by element-wise multiplication between the value obtained from a tanh function of c_t and the output of a sigmoid layer (i.e., the output gate), o_t . Through the cooperation between the memory cell and the gates, LSTM is endowed with a powerful ability to predict time series with long-term dependences.

Since the invention of LSTM, a number of scholars have proposed several improvements with respect to its original architecture. Greff *et al.* [12] evaluated the aforementioned conventional LSTM and eight different variants thereof (e.g., Gated Recurrent Unit (GRU) [13]) on three benchmark problems—TIMIT, IAM Online and JSB Chorales. Each variant differs from the conventional LSTM by a single and simple change. They found that the conventional LSTM architecture performs well on the three datasets, and none of the eight investigated modifications significantly improve the performance.

III RANDOM CONNECTIVITY FOR LSTM

The conventional LSTM (including its variants) follows the classical pattern that the neurons in each memory block are fully connected and this connectivity cannot be changed manually. However, it has been found that for certain functional connectivity in neural microcircuits, random topology formation of synapses plays a key role and can provide a sufficient foundation for specific functional connectivity to emerge in local neural microcircuits [14]. This discovery is different from the conventional cases where neural connectivity is considered to be more heuristic so that neurons need to be connected in a more fully organized manner. It raises a fundamental question as to whether a strategy of forming more random neural connectivity, like in the human brain, might yield potential benefits to LSTM's performance and efficiency. With this conjecture, we built up the RCLSTM model.

In RCLSTM, neurons are randomly connected rather than being fully connected as in LSTM. Actually, the trainable parameters in LSTM only exist between the input part—the combination of the input of the current moment (i.e. x_t) and the output of the previous moment (i.e. h_{t-1}), and the functional part—the combination of the gate layers and the input update layer. Therefore, the LSTM architecture can be further depicted in Fig. 2. In our approach, whether the LSTM neurons are connected or not can be determined by certain randomness. Therefore, we use dashed lines to denote that the neural connections can be added or omitted, as depicted in the upper part of Fig. 2. If the neurons are fully connected, then it becomes a standard LSTM model. On the other hand, if the neurons are randomly connected according to some rules (which are covered in detail below), then an RCLSTM model is created. The lower right part of Fig. 2 shows an example RCLSTM structure in which the neural connections are randomly sparse, unlike the LSTM model. The fundamental difference between RCLSTM and LSTM is illustrated in Fig. 2, so let us move to the implementation strategy of randomly connecting neurons.

First, we attach a probability value to each pair of neurons that are connected by a dashed line in the upper part of Fig. 2. The probability values can obey arbitrary statistical distributions, and we choose uniform distribution in our simulations given its computational efficiency. The probability value indicates the tendency that the corresponding pair of neurons will be connected. Then we assume all neurons are connected with the same probability and carefully set a threshold to determine the percentage of connected neurons. If the probability values are greater than the threshold, the corresponding pairs of neurons are connected. Otherwise, they are prohibited from being connected. This process can be visualized as turning dashed lines into solid lines, as shown in the right-hand transformation of Fig. 2. Therefore, the RCLSTM structure can cre-

ate some sparsity, considerably decreasing the total number of involved parameters to be trained and reducing the computational loads of the whole RCLSTM network.

IV NUMERICAL SIMULATIONS FOR TRAFFIC AND MOBILITY PREDICTION

In this section, we focus on verifying the performance of the proposed RCLSTM model on traffic prediction and user-location forecasting. In particular, we construct a three-layer RNN whose recurrent neurons are all the newly designed RCLSTM memory blocks (for the sake of simplicity, this model is called the RCLSTM network in the following statement). The corresponding expanded form is described in Fig. 3. Because the purpose of our simulations is to predict the value at the next moment given some historical data, the input data are from y_1 to y_T where T denotes the length of the input sequences, and the output of the RCLSTM network is a prediction of the actual value at the next moment denoted as \hat{y}_{T+1} . First of all, we take advantage of the RCLSTM network to predict traffic and user mobility, particularly compare the prediction accuracy of the RCLSTM network with that of other algorithms or models. Then, we adjust the number of training data samples and the length of input sequences to investigate the influence of these factors on the prediction accuracy of the RCLSTM network.

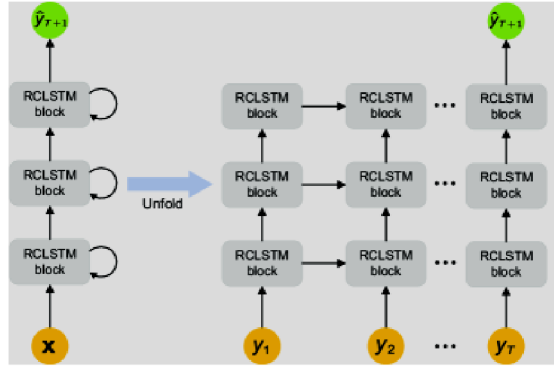


Figure 3: The designed RCLSTM network for simulations.

IV-A Data Description and Processing

We evaluate the model's performance on traffic prediction depending upon real traffic data from a link in the GÉANT backbone networks [9]. GÉANT is a pan-European data source for the research and education communities. These traffic data are sampled every 15 minute during a 4-month period and the unit for data points is Kbps. In this study, we selected 10772 traffic data points from 2005-01-01 00:00AM to 2005-04-30 00:00AM. The raw data are so uneven in numerical size that there are about three orders of magnitude difference between the maximum and minimum. To reduce the numerical unevenness, we first take the logarithm to base 10 of the raw data, then carry out a normalization process according to $\frac{x - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$,

where \mathbf{x} is the vector after taking the logarithm of the raw data, $\min(\mathbf{x})$ and $\max(\mathbf{x})$ denote the minimum and maximum value of \mathbf{x} , respectively. Through this process, the raw data are limited to a range between 0 and 1, which makes the training phase of ANN-based models converge faster and effectively avoid the bad local optimal solution [11]. Real-time prediction of data traffic requires continuous data input and learning. Hence, we introduce a notion of sliding window, which indicates a fixed number of previous timeslots to learn and then predict the current data traffic. Finally, we split the processed data into two sets (i.e., a training set and a test set). The training set is used to train the RCLSTM network, and the test set is used to evaluate its prediction accuracy.

The other dataset to evaluate the capability of the RCLSTM model comes from reference [10], which contains the location history of several mobile users together with manually defined important places for every person, and is thus used for user-mobility prediction. The data for every person are taken from the Android location history service, and mobile users are asked to mark the important places that are further constructed

as polygons on the map. The dataset contains four attributes, i.e., date-time, latitude, longitude, and assigned location ID, within which the last item is used to map the user's location to one of the geographical polygons. These data are collected in 1-hour intervals from 2015-08-06 to 2015-11-04. In this article, we use location data of five users. First, we encode the location ID to one-hot code. For example, suppose that in User A's trajectory, the number of different location IDs marked by User A is m . Then we map these different location IDs to the Numbers 1 to m , and use them as new location IDs. If the newly current location ID is i , we construct an m -dimensional vector where only the element with index i is one and the others are all zero. After transforming all the raw data to one-hot vectors, we use the sliding window to slice the processed data and finally split them into training set and test set, which follows the same procedure as mentioned earlier in the traffic data processing.

IV-B Evaluation Metrics

To evaluate the performance of our proposed RCLSTM model on traffic prediction, Root Mean Square Error (RMSE) is applied to estimate the prediction accuracy. RMSE measures the squared root of the mean of the deviation squares, which quantifies the difference between the predicted values and the actual ones. The RMSE can be expressed as $RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$, where y_i is the actual value, \hat{y}_i is the predicted value and N represents the total number of predicted traffic data.

On the other hand, the evaluation metric for human mobility prediction is the accuracy level, which indicates the percentage of the correct location predictions. In our study, the accuracy level is defined as $Acc = \frac{\sum_{i=1}^N \mathbf{1}(y_i = \hat{y}_i)}{N}$, where y_i is the actual value, \hat{y}_i is the predicted value, N represents the total number of predicted locations and $\mathbf{1}(y_i = \hat{y}_i)$ is the indicator function that equals to 1 if $y_i = \hat{y}_i$ or 0 if $y_i \neq \hat{y}_i$.

IV-C Testing Results and Analyses

IV-C1 Traffic Prediction

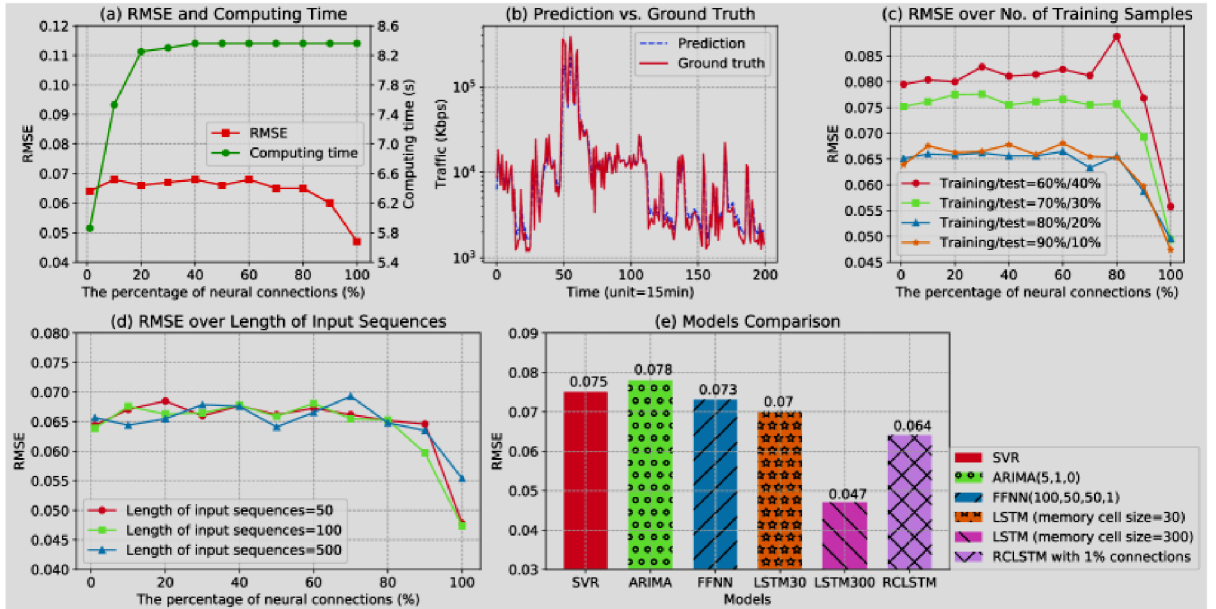


Figure 4: Results of traffic prediction using RCLSTM network: (a) RMSE and computing time of RCLSTMs ; (b) predicted traffic data and actual traffic data; (c) RMSE of RCLSTMs over different number of training samples; (d) RMSE of RCLSTMs over different length of input sequences; (e) comparing RCLSTM with SVR, ARIMA, FFNN and LSTM.

Fig. 4(a) reveals the RMSE and the computing time under different percentages of neural connectivity in the RCLSTM model (note that 100% connectivity means the fully-connected LSTM model). Notably, the probability of neural connections obeys a uniform distribution between 0 and 1. In addition, the size of the RCLSTM's memory cell is set at 300, the ratio between the number of training samples and the number of test samples is set at 9:1, and the length of input traffic sequences is 100. Fig. 4(a) shows that the RMSE of the RCLSTM model is slightly larger than that of the LSTM model, but the RCLSTM with very sparse neural connections (i.e. 1%) reduces the computing time by around 30% compared with the baseline LSTM. In addition, the computing time almost stops increasing when the percentage of neural connections is larger than 20%, which is because the method we use for accelerating calculation only works efficiently on extremely sparse matrices. Fig. 4(b) intuitively illustrates the actual and predicted traffic values. It can be observed from the subfigure that the predicted values can match the variation trend and features of the actual values very well. Therefore, the simulation results indicate that the RCLSTM model can yield acceptable prediction capability, and effectively decrease the computational loads and complexity.

Then we investigate how the predictive capability of the RCLSTM model is influenced by the number of training samples and the length of input traffic sequences. First we train the models with 90%, 80%, 70% and 60% of the processed data, respectively, while fixing both the size of the memory cell (set at 300) and the length of the input sequences (set at 100), and test the trained models with the remaining data. Then we vary the length of the input sequences from 50 to 500 while keeping the values of the other hyperparameters fixed. The simulation results are shown in Fig. 4(c) and (d). It can be observed from Fig. 4(c) that RCLSTM models are more sensitive to the number of training samples than LSTM, because when the number of training samples increases, the RMSEs of the RCLSTM models vary more significantly than that of the LSTM model. Fig. 4(d) gives the related results and shows that RCLSTM models are less susceptible to the length of the input sequences than LSTM model.

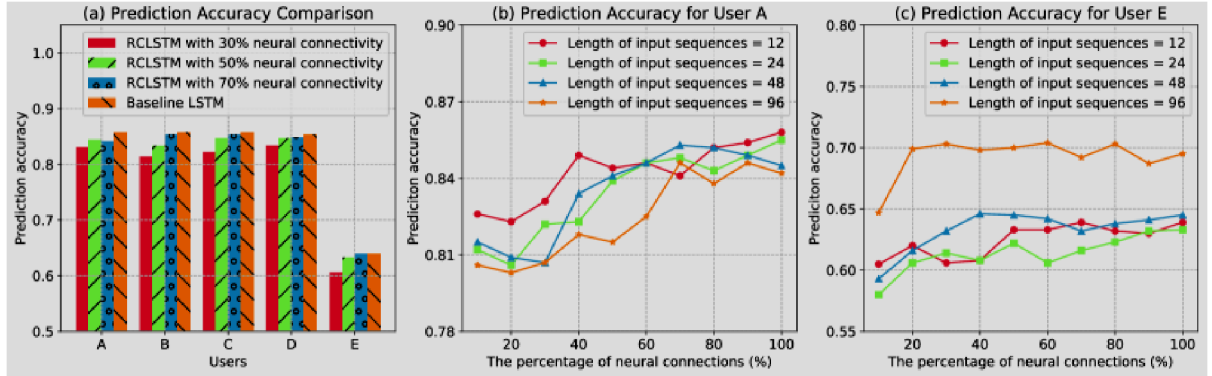


Figure 5: Results of human mobility prediction using RCLSTM: a) prediction accuracy of RCLSTMs for different users; b) prediction accuracy of RCLSTMs over different length of input sequences for User A; c) performance of RCLSTMs over different length of input sequences for User E.

Finally, we compare the RCLSTM with three well-known prediction techniques—SVR, ARIMA, and FFNN. The hyper-parameters of these algorithms are as follows:

- SVR: The number of input features is 100, the kernel is a Radial Basis Function (RBF) and the tolerance for the stopping criterion is 0.001.
- ARIMA(p, d, q): The number of autoregressive terms (i.e. p) is 5, the number of nonseasonal differences needed for stationarity (i.e. d) is 1, and the number of lagged forecast errors in the prediction equation (i.e. q) is 0.
- FFNN: The number of input features is 100 and the numbers of neurons in both the first hidden layer and the second hidden layer are 50.

In addition, since the LSTM with a memory cell size of 30 has almost as many trainable parameters as the RCLSTM with a memory cell size of 300 and 1% neural connections, we put it into the comparison list as well.

The simulation results are shown in Fig. 4(e), which reveals that LSTM with a memory cell size of 300 performs much better than the others, followed by the RCLSTM with the memory cell size of 300 and 1% neural connections. Interestingly, the RCLSTM model performs better than the LSTM with the memory cell size of 30, which is probably due to a degree of overfitting that exists in the latter [11].

IV-C2 Human Mobility Prediction

The results of user-mobility prediction with the RCLSTM model are shown in Fig. 5. Fig. 5(a) shows the respective prediction accuracy for the five users with the RCLSTM model, where the probability of neural connections obeys a uniform distribution between 0 and 1. In addition, the size of the memory cell is 150, the length of input sequences is 12, and the training samples account for 90% of the processed data. There is a slight difference in the prediction results for different users. For example, Users A and D are both university students with a part-time job, and thus they almost follow the same behavioral pattern in school or work, which results in highly expected predictability. On the other hand, User E is running his own business and is more likely to travel a lot with unfixed schedule, consequently having low expected predictability [10]. Although the prediction accuracy of the RCLSTM model is not as good as that of the LSTM model, the RCLSTM model with high sparsity of neural connections can compute faster than the LSTM, similar to the traffic prediction scenario.

In order to further investigate the capability of the RCLSTM model to characterize long-term dependencies of user mobility, we carry out simulations with the different length of input sequence data for Users A and E, consistent with the scenario of traffic prediction. The results are shown in Fig. 5(b) and (c), respectively. It can be observed from the subfigures that the RCLSTM models can catch up with LSTM in terms of prediction accuracy, regardless of the length of the input sequences. Remarkably, for User A, increasing the length of input sequences scarcely affects the prediction accuracy, whereas for User E, the prediction accuracy improves when the length of the input sequences increases. From the perspective of data analysis, the mobility data of User E is more irregular than that of User A, so both the RCLSTM model and the baseline LSTM model need more input data to find the regular movement patterns of User E.

Remark. *The RCLSTM model shows promise for manifesting strong traffic and user-mobility prediction capabilities while reduces the number of parameters to be trained, which in effect decreases the computational load and complexity.*

V CONCLUSION

In this article, we have addressed the importance of leveraging deep learning for time series prediction. In particular, we have reinvestigated the issues of traffic prediction and user-mobility forecasting with deep learning and proposed a new model named RCLSTM by revolutionarily redesigning the conventional LSTM model. The basic idea behind the RCLSTM model is to construct neural networks by forming and realizing a random sparse graph. We have checked the effectiveness of the RCLSTM model by predicting the dynamics of traffic and user locations through various temporal scales. In traffic prediction, we have demonstrated that the RCLSTM model with 1% neural connections reduces the computing complexity by 30% compared with the standard LSTM model. Although the characteristic of sparse neural connections may cause a degradation of approximately 25% in prediction accuracy, the RCLSTM model still outperforms SVR, ARIMA, FFNN, even the LSTM model with the same number of parameters. In addition, along with the adjustment of the length of input sequences and the number of training samples, the RCLSTM models fluctuate in line with the LSTM model in terms of prediction accuracy. In summary, it can be expected the RCLSTM model with lower computing costs and satisfactory performance will play an essential role in time series prediction in the future intelligent telecommunication networks.

REFERENCE

- [1] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, and H. Zhang, "Intelligent 5G: When Cellular Networks Meet Artificial Intelligence," *IEEE Wireless Communications*, vol. 24, no. 5, Oct 2017, pp. 175–183.

- [2] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal Modeling and Prediction in Cellular Networks: A Big Data Enabled Deep Learning Approach," *Proc. 36th International Conference on Computer Communications (INFOCOM)*, Atlanta, GA, USA, May 2017, pp. 1–9.
- [3] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," *Proc. 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 961–971.
- [4] N. I. Sapankevych and R. Sankar, "Time Series Prediction Using Support Vector Machines: A Survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, May 2009, pp. 24–38.
- [5] W.-C. Hong, "Application of Seasonal SVR with Chaotic Immune Algorithm in Traffic Flow Forecasting," *Neural Computing and Applications*, vol. 21, no. 3, Apr 2012, pp. 583–593.
- [6] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Learning Tools and Techniques*, Morgan Kaufmann, Burlington, MA 2016,
<https://www.amazon.com/exec/obidos/ASIN/0128042915/de>
accessed 10/11/2018.
- [7] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, 1997, pp. 1735–1780.
- [8] M. J. Shafiee, P. Siva, and A. Wong, "Stochasticnet: Forming Deep Neural Networks via Stochastic Connectivity," *IEEE Access*, vol. 4, 2016, pp. 1915–1924.
- [9] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing Public Intradomain Traffic Matrices to the Research Community," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, Jan 2006, pp. 83–86.
- [10] J. Tkačík and P. Kordík, "Neural Turing Machine for Sequential Learning of Human Mobility Patterns," *Proc. 2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Canada, July 2016, pp. 2790–2797.
- [11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, 2015, p. 436.
- [12] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, Oct 2017, pp. 2222–2232.
- [13] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *CoRR*, vol. abs/1412.3555, 2014.

S. L. Hill, Y. Wang, I. Riachi, F. Schürmann, and
H. Markram, “Statistical Connectivity Provides a
Sufficient Foundation for Specific Functional
Connectivity in Neocortical Neural Microcircuits,”
Proceedings of the National Academy of Sciences, vol.
109, no. 42, 2012, pp. E2885–E2894.

