A Ride-Hailing Supply and Demand Prediction Method Based on Recurrent Neural Networks (Postprint)

Authors: An Lei, Zhao Shuliang, Yongliang Wu, Chen Runzi, Jiaxing Li

Date: 2018-05-02T00:00:00+00:00

Abstract

Utilizing real-world ride-hailing order data as the data source, combined with the TensorFlow deep learning framework, this study employs recurrent neural networks (RNN) to predict order demand for ride-hailing services at specific future times and locations. An improved LSTM RNN (Long Short-Term Memory Recurrent Neural Network) model is proposed, which, after optimization and training, can effectively predict supply and demand volumes for ride-hailing services at specific times and locations in the future. Visual analysis of the data source is conducted to eliminate interference from irrelevant data sources, forming the foundation for designing simulation experiments. The simulation experiments demonstrate that the model achieves higher accuracy than back-propagation neural networks (BPNN), decision tree regression (DTR), support vector regression (SVR), and random walk (RW) models. Moreover, it exhibits favorable memory capabilities for historical data with varying intervals and strong generalization ability on test data.

Full Text

Preamble

Prediction Method of Supply and Demand for Online Car-Hailing Services Based on Recurrent Neural Networks

An Lei^{a,b}, Zhao Shuliang^{a,b}, Wu Yongliang^{a,b}, Chen Runzi^{a,b}, Li Jiaxing^{a,b}

^{a}College of Mathematics & Information Science; ^{b}Hebei Key Laboratory of Computational Mathematics & Applications, Hebei Normal University, Shijiazhuang 050024, China

Abstract: Using real-world data sources such as online car-hailing order records and the TensorFlow deep learning framework, this study employs recurrent neural networks to predict order demand for online car-hailing services at specific times and locations. We propose an improved Long Short-Term Memory Recurrent Neural Network (LSTM RNN) model that, after optimization and training, can effectively predict supply and demand quantities. Through visual analysis of the data source to exclude irrelevant interference, we design simulation experiments based on these insights. Experimental results demonstrate that our model achieves higher accuracy than Back Propagation Neural Networks (BPNN), Decision Tree Regression (DTR), Support Vector Regression (SVR), and Random Walk (RW) models. Additionally, the proposed model exhibits strong memory capabilities for historical data with varying time intervals and demonstrates robust generalization ability on test data.

Keywords: long short-term memory recurrent neural networks; online car-hailing data; traffic optimization scheduling; TensorFlow; deep learning

0 Introduction

Deep learning has made increasingly significant contributions to artificial intelligence research, particularly in speech recognition [**?**] and image recognition [**?**]. Researchers have also achieved promising results applying deep learning methods in other domains, such as the Go AI AlphaGo [**?**]. Deep learning algorithms can be broadly categorized into four types: deep neural networks, convolutional neural networks, recurrent neural networks, and reinforcement learning [**?**]. With the rapid development of these methods, numerous deep learning frameworks have gained attention, including TensorFlow [**?**], Caffe [**?**], Keras, CNTK, and MXNet.

TensorFlow, officially open-sourced by Google on November 9, 2015, is a computational framework whose model can effectively support deep learning algorithms including deep neural networks, and the system demonstrates high stability [**?**]. TensorFlow has attracted attention from both academia and industry. For example, Geoffrey Hinton et al. utilized the TensorFlow framework for capsule network experiments, combining it with a proposed dynamic routing method to achieve promising results on overlapping digit recognition problems compared to standard convolutional neural networks [**?**]. Wongsuphasawat et al. proposed a TensorFlow-based dataflow graph visualization method for deep learning models [**?**]. Uber implemented autonomous driving technology based on TensorFlow' s AlexNet deep learning model [**?**], while companies such as Twitter, JD.com, and Xiaomi also employ TensorFlow. In summary, this paper uses the TensorFlow deep learning framework for our experiments.

The supply-demand dynamics of online car-hailing services relate to order volume fluctuations and are also associated with factors such as weather, regional infrastructure, and traffic conditions. Consequently, supply-demand variations

exhibit high nonlinearity and randomness, making accurate prediction of supply-demand gaps challenging. Artificial Neural Networks (ANN) possess the capability to solve nonlinear and stochastic problems [**?**]. Data in the source constitutes time series data, which Recurrent Neural Networks (RNN) can predict [**?**] (time dependencies in the data source are analyzed in detail below). In the test set, data from the half-hour preceding the target time slice is known; however, supply-demand fluctuations in the next time slice relate not only to short-term data but also to data from more distant time periods. Therefore, we propose using Long Short-Term Memory Recurrent Neural Networks (LSTM RNN) as the solution for supply-demand prediction.

Traffic optimization scheduling represents a crucial component in smart city development. Traditional traffic flow research focuses on two aspects: first, using taxi GPS location and speed information to reflect road congestion levels; second, employing historical taxi data or data collected from road monitoring stations for mining through parametric or non-parametric methods. Parametric methods such as the Kalman filtering model [**?**] were frequently applied in early traffic flow prediction research. These model structures are based on theoretical assumptions, with parameters calculated from empirical data. The most widely applied parametric method is the Autoregressive Integrated Moving Average (ARIMA) model, which assumes stationary traffic states. ARIMA is denoted as ARIMA(p,d,q), where p, d, and q represent three parameters. Levin and Tsao used this method to predict highway traffic flow, concluding that ARIMA(0,1,1) was the most effective model [**?**].

However, traffic flow exhibits nonlinear and stochastic characteristics, making accurate and effective prediction difficult for traditional models. In artificial intelligence, methods such as SVM or SVR demonstrate stronger capability in capturing features from irregular data [**?**]. In recent years, non-parametric methods have gained attention. Support Vector Machine (SVM) essentially maps data through nonlinear relationships into high-dimensional space for linear regression. Castro-Neto et al. used OL-SVR to predict traffic flow under both typical and atypical conditions (e.g., holidays and traffic accidents) [**?**]. ANN is also widely applied in traffic flow prediction, capable of handling high-dimensional data with complex model structures while exhibiting strong generalization and learning abilities [**?**]. Vlahogianni et al. optimized neural networks through genetic algorithms for short-term traffic flow prediction [**?**]. Yu et al. used BP neural networks to monitor traffic congestion [**?**]. Chai et al. combined wavelet analysis with neural networks for short-term traffic flow prediction [**?**]. Chen used RBF neural networks for traffic flow prediction, proposing three algorithms to optimize RBF weights and thresholds [**?**]. Wang et al. applied BP neural networks to predict bus traffic and proposed optimization strategies [**?**]. Yu et al. proposed an improved artificial bee colony algorithm based on RBF neural networks for traffic prediction [**?**]. Wang et al. proposed the DeepSD deep neural network model for predicting online car-hailing supply and demand [**?**].

Nevertheless, existing non-parametric methods require pre-defined training data

lengths that cannot be changed, and they seldom consider information such as weather, traffic congestion, and regional infrastructure. Their data sources typically come from monitoring stations or Intelligent Transportation Systems (ITS), whereas this paper utilizes online car-hailing data.

This research addresses the aforementioned problems. To improve prediction accuracy, we propose a model called Long Short-Term Memory Recurrent Neural Network (LSTM RNN), which can more effectively capture the nonlinearity and randomness in data sources. Through memory blocks, it overcomes the decay problem in error backpropagation while satisfying the time series dependencies of the data source. Moreover, LSTM RNN achieves higher prediction accuracy on test sets.

The data source is analyzed from both spatial and temporal perspectives. Spatially, a city is divided into n non-overlapping square regions represented as set $D = \{d_i | i \in [1, 58]\}$. Temporally, 24 hours are divided into 144 ten-minute time slices represented as set $T = \{t_i | i \in [2350, ..., 0010, 0000\}$, with each t_i corresponding to a time_d. Some scholars adopt 15-minute intervals [**?**]. Based on these spatial and temporal dimensions, for region d_i and time slice t_j, we define the supply-demand gap gap_{d_i,t_j} as the number of orders without driver acceptance.

Supply-demand prediction constitutes part of traffic flow forecasting. Short-term traffic flow prediction methods fall into two categories: parametric and non-parametric. The data source comprises time series data. LSTM, proposed by Hochreiter and Schmidhuber [**?**], enables input time series data to be memorized through a cyclic structure that passes historical information forward. Therefore, the definition of state vectors represents one of the keys to RNN. However, as the cycle progresses, information from earlier moments' influence on current supply-demand gaps diminishes—the vanishing gradient problem [**?**]. In the data source visualization analysis in Section 4, the time series data length is $DataLenth = 33 \times 1223 \times 144$. Historical information affecting the next time slice's supply-demand gap varies in distance from the current moment, including data from recent time slices as well as data from the same time period on previous days and surrounding time slices.

1 Long Short-Term Memory Recurrent Neural Network (LSTM RNN)

The primary purpose of RNN is to process and predict sequence data, utilizing historical information to help solve current problems. Consequently, it can leverage information that traditional network structures cannot capture. In the data source, factors influencing supply-demand gap size in a given time slice include not only order demand and supply but also traffic, weather, and day of week, among others.

The LSTM structure comprises one input layer, one recurrent body structure, and one output layer, as shown in [Figure 1: see original paper]. The recurrent

body contains three gates: forget gate, input gate, and output gate. A "gate" refers to a combination of a neural network with sigmoid activation function and a bitwise multiplication operation. The sigmoid-activated neural network outputs a value between 0 and 1, describing whether current input can pass through the structure. Gate structures mitigate the vanishing gradient problem. For example, in the experiments in Section 4, when time series data (3456 dimensions) is input at the current moment, if the input gate closes (sigmoid neural network output layer outputs 0), the current input will not affect the current state.

We define the time series data set as $X = \{x_1, x_2, ..., x_{state}\}$, state vector $h_i \in \{h_1, h_2, ..., h_i\}$, and use MAE as the loss function. The LSTM RNN algorithm, given known supply-demand gap values $Y = \{y_{date,time}\}$, where date range is $\{0223, ..., 0224, 0317\}$ and time range is $\{0000, 0010, ..., 2350\}$, proceeds as follows:

Time series data is divided into sample data according to truncation length, where supply-demand values from the preceding TIMESTEPS time slices serve as samples, and the target time slice' s supply-demand value serves as the label, forming a complete individual sample, as shown in Algorithm 1.

Algorithm 1 Sample Set Construction Algorithm Input: Time series data seq, truncation length TIMESTEPS. Output: Sample set X, sample label set Y.

```
def generate_data(seq, TIMESTEPS):
    for i in range(len(seq) - TIMESTEPS):
        X.append([seq[i:i+TIMESTEPS]])
        Y.append([seq[i+TIMESTEPS]])
    return X, Y
```

Forward propagation results are obtained through the following equations:

The hidden layer input at time t is:

$$\hat{h}_t = W_{hh} h_{t-1} + W_{hx} x_t + b_h \tag{1}$$

The output is:

$$gap'_{time,date} = W_{output} h_t + b_{output} \tag{2}$$

where the initial hidden state $h_0 = (0, 0, ..., 0)$ and input $x_t = gap_{time,date,t}$.

The forget gate function f_t, input gate function i_t, and cell activation function $\tilde{c}_t$ are:

$$f_t = \sigma(W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f) \tag{3}$$

$$i_t = \sigma(W_{xi} x_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i) \tag{4}$$

$$\tilde{c}_t = \sigma(W_{xc} x_t + W_{hc} h_{t-1} + b_c) \odot f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{5}$$

where σ is the sigmoid function with range $[0, 1]$ and tanh has range $[-2, 2]$.

The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

The output gate o_t is defined as:

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{6}$$

The state h_t is influenced by the output gate:

$$h_t = \tanh(c_t) \odot o_t \tag{7}$$

To make the LSTM RNN model more robust, prevent overfitting on training data, and improve model accuracy on test data (i.e., enhance generalization capability), we add a binary classification function to the cell activation function (5) that obtains integer 1 with probability p and integer 0 with probability $1-p$ [**?**]. This controls whether the cell activation value takes effect. If the function value is 1, the activation is valid; if 0, it becomes invalid:

$$c_t = p_{Binary} \odot (\sigma(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \odot f_t \odot c_{t-1} + i_t \odot \tilde{c}_t) \tag{8}$$

We construct the LSTM RNN single-layer network structure as shown in Algorithm 2(2), construct a binary classification function for single-layer structure optimization as shown in Algorithm 2(3), concatenate NUM_LAYERS optimized single-layer structures into a complete LSTM RNN structure as shown in Algorithm 2(4), then compute the output through forward propagation, and finally obtain predictions and loss values through a fully connected layer, as shown in Algorithm 2(5) and (6).

Algorithm 2 LSTM RNN Construction Algorithm Input: Sample set X, sample label set Y, number of hidden nodes HIDDEN_SIZE, number of hidden layers NUM_LAYERS. Output: Prediction result prediction, loss function loss.

```
def lstm_model(X, Y, HIDDEN_SIZE, NUM_LAYERS):
    lstm_cell ← BasicLSTMCell(HIDDEN_SIZE)
    drop_lstm ← Dropout(lstm_cell, output_keep_prob)
    cell ← MultiRNNCell([drop_lstm] * NUM_LAYERS)
    Output ← rnn(cell, X)
    prediction, loss ← regression(output, Y)
    return prediction, loss
```

The training set data source is detailed in Section 4. We train the LSTM RNN model using this data through backpropagation to minimize the loss function, defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |gap_{d_i,t_{date}} - gap'_{d_i,t_{date}}| \tag{9}$$

Given data for a certain region d_i on day $date$, we predict the supply-demand gap $gap'_{d_i,t_j,date}$ for time t_j.

Algorithm 1 has time complexity O(1). Algorithm 2's time complexity comprises four components:

$$O(WO) + O(CSI) + O(HIK) + O(CSK) + O(HO) \tag{10}$$

where K represents output units, H hidden units, C memory units, S memory unit size, and I feedforward connections to memory and gate units. LSTM RNN time complexity does not depend on network structure or input time series length [**?**], with overall complexity O(W), where W represents the number of weights, making LSTM RNN efficient.

All Turing machines can be simulated by fully connected recurrent networks built on neurons with sigmoid activation functions [**?**]. Turing machines can compute any computable function. Since the activation function (6) is a sigmoid function, the computational model proposed in this paper can theoretically simulate supply-demand gap prediction functions.

3.1 Data Source Visualization and Analysis

Visual analysis of the data source aims to reasonably divide training and test sets. Based on this, we classify the data source to train models and compare prediction accuracy before and after classification. Without classification, LSTM RNN achieved an RMSE of 58.171 on the test set, greater than the error of 43.189 after classification, demonstrating that classified data sources yield better training results and lower test set errors with higher prediction accuracy.

Data originates from the first Di-Tech algorithm competition, with all data being authentic. The training set is 1.46GB, including continuous 24-day data from a city in 2016. We select partial data for experiments. The order information table, weather information table, and POI information table are directly obtained from the database, while the region definition table and congestion information table are derived from other database tables.

The order information table contains fields: order_ID, price, driver_ID, passenger_ID, start_district_hash, time, and dest_district_hash. Based on 10-minute intervals, 24 hours are divided into 144 time slices. Within each time slice, orders are assigned to time slices by comparing the time stamp field in the order information table.

Define date set $D_{date} = \{0223, ..., 0224, 0317\}$ and demand $Demand_{date}$. In [Figure 2: see original paper], $Demand_{0312} = 444387$ represents the order demand for a city on March 12, 2016, reaching a maximum compared to surrounding days, with supply-demand gap $Gap_{0312} = 82662$ also reaching a maximum, while the maximum supply-demand gap $Gap_{0308} = 84130$ appears on Tuesday, March 8, 2016.

Define order count $C_{i,j}^t$ where i represents departure district ID ($i \in [1, 58]$) and j represents destination district ID ($j \in [1, 58]$), i.e., $C_{i,j}^t$ = order count. In [Figure 3: see original paper], the set $\{C_{44,j}^t | j \in [1, 58]\}$ shows large values, indicating high order demand for trips originating from district 44, where $C_{44,44}^t = 12554$ represents the highest intra-district order volume in that area on that day. In the POI information table, district 44 has the most facility categories and largest facility quantity. The set $\{C_{i,j}^t | i \in [1, 58], j \in [1, 58]\}$ occupies most total order volume, particularly concentrated in $\{C_{i,j}^t | i \in [1, 22], j \in [1, 22]\}$, while other regions have smaller order volumes, with some areas having zero orders. The set $\{C_{i,j}^t | i \in [1, 58], j \in [1, 58]\}$ generally shows large values, indicating high demand for intra-district trips.

On March 1, 2016, from 08:30-08:40, order demand reached a maximum of 4773, with supply-demand gap $Gap_{0830} = 895$ also reaching a maximum. Another demand peak occurred at 17:30 with $Demand_{1730} = 4141$, while $Gap_{0000} = 470$ and $Demand_{2350} = 12942$. After midnight, demand decreases and the supply-demand gap remains stable and low. Examining the congestion information table for the same time period shows fewer congested road segments, with some recovery compared to the previous moment.

Define gap sets: $G_{time}^1 = \{Gap_{time} | time \in [0720, 0930]\}$, $G_{time}^2 = \{Gap_{time} | time \in [1520, 1610]\}$, $G_{time}^3 = \{Gap_{time} | time \in [1720, 1820]\}$, and $G_{time}^4 = \{Gap_{time} | time \in [2100, 2200]\}$. These four sets generally show large gap values, indicating insufficient supply in these time periods.

[Figure 5: see original paper] shows that over 8 days from February 23 to March 1, 2016, the maximum supply-demand gap occurred on February 23 at 08:00-08:10 (a Tuesday). Peak demand increases also appeared Monday through Friday, while no peaks occurred on the weekend days of February 27-28, especially on Sunday February 28 when the gap was particularly small. During these weekend days, the supply-demand gap remained growing and stable for extended periods, decreasing near 17:20 and increasing at 18:40. Notably, on February 24 at 15:30, the gap increased from 287 to 1372 within 10 minutes. Order supply-demand changes follow a daily cycle, with clear differences between weekends and weekdays. By predicting future supply-demand gaps in specific regions at specific times, we can proactively implement dispatch measures to optimize scheduling and alleviate travel pressure.

On February 23, 2016, supply-demand gap values across 58 regions and 144 time slices are shown in [Figure 6: see original paper], where region 44 exhibits large gap values during several time periods, with $Gap_{44,1650,0223} = 244$. Different regions show varying intra-day supply-demand patterns, making regional information a crucial factor during model training. Different regions exhibit certain patterns in supply-demand changes over consecutive days, so time series data selects data from the same region over consecutive days for training.

3.2 Experimental Design

We select data from the first 23 days as the training set and the final day' s data as the test set. The training set consists of 23 days of data, with weekend data excluded based on the target prediction time. To compare with other methods, we test four additional approaches: BP Neural Network, Support Vector Regression (SVR), Decision Tree Regression (DTR), and Random Walk (RW). BPNN is a fully connected neural network and the most fundamental ANN method. As a deep learning method like LSTM RNN, comparing them reveals how network structure affects model performance. Support vector machines perform well in both classification and regression, with performance varying by kernel function (linear, polynomial, Gaussian, or sigmoid). Decision tree regression is a greedy algorithm performing recursive binary partitioning in feature space. Random Walk is a simple method that uses the current supply-demand gap to predict the next time slice' s gap.

We implement the LSTM RNN model using Python 2.7, TensorFlow 0.11.0, Protobuf 3.4.0, Scikit-learn 0.19.0, and Scipy 0.17.0. To compare LSTM RNN' s predictive capability against the other four methods, we compare their Root Mean Square Error (RMSE), which measures dispersion between true and predicted values [**?**]:

$$RMSE(Gap, Gap') = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Gap_i - Gap'_i)^2} \tag{11}$$

Algorithm 3 RMSE Calculation Algorithm Input: Training iterations TRAINING_STEPS, training set seq_train, test set seq_test. Output: Root mean square error rmse.

```
train_X, train_Y ← generate_data(seq_train, TIMESTEPS)
test_X, test_Y ← generate_data(seq_test, TIMESTEPS)
TrainedModel ← train(train_X, train_Y, TRAINING_STEPS)
predicted ← TrainedModel.predict(test_X)
rmse ← sqrt((predicted - test_Y) ** 2).mean()
```

Time series data is selected from the training set and processed through Algorithm 1 to obtain training sample and label sets. Similarly, test sample and label sets are obtained (Algorithm 3(1) and (2)). After TRAINING_STEPS iterations of training (Algorithm 3(3)), the trained model predicts test data (Algorithm 3(4)). Finally, RMSE between predictions and true values is calculated.

3.3 Prediction Accuracy

[Figure 7: see original paper] shows actual and predicted supply-demand values over 24 hours in region 44 on March 17, 2016, visually demonstrating that the predicted curve fits the actual curve well. March 17 was a Thursday, so weekend data (6 days) was excluded from the training set. Model parameters

and RMSE comparisons are shown in . Note that listed RMSE values represent the minimum obtained from 10 experiments under identical conditions for each method. Prediction results for the other four methods are shown in [Figure 8: see original paper].

Comparisons reveal BPNN' s RMSE at 61.218, SVR at 92.217, DTR at 98.992, and RW at 256.393. LSTM RNN' s error across 10 experiments had a maximum of 46.377, indicating small error variation and stable prediction performance on the test set. BPNN' s maximum error reached 95.221 across 10 experiments, showing large fluctuations and unstable performance. SVR' s error varied minimally, with RMSE stable around 92, indicating high prediction stability. DTR' s maximum error was 99.339, with moderate fluctuation but less than BPNN. RW showed large error variation, demonstrating high randomness and large errors when predicting the next moment based only on the previous time slice.

LSTM RNN structure directly impacts predictive capability, particularly the number of hidden layers and nodes per layer. By setting different values to find the optimal structure, we observe the effect of node count on RMSE with 2 hidden layers. As shown in [Figure 9: see original paper], RMSE is large with few hidden nodes, decreasing as node count increases. RMSE reaches its minimum at 68 hidden nodes, stabilizing around 50 with further increases.

3.4 Relationship Between Truncation Length and RMSE

Truncation length refers to the number of time slices preceding the target prediction slice. For example, a 30-minute truncation length equals 3 (with 10-minute time slices). Under identical model structures, different truncation lengths significantly impact prediction accuracy, as shown in [Figure 10: see original paper].

The figure shows minimum RMSE of 43.189 at truncation length 2, meaning data from 20 minutes before the prediction time slice has the greatest impact on accuracy. The maximum truncation length shown is 18, representing 3 hours of historical data. Experiments show RMSE of 44.645 at length 1, 54.687 at length 3, and 44.112 at length 4. When truncation length exceeds 4, RMSE increases progressively. LSTM RNN' s ability to memorize historical data of varying lengths represents one of its key advantages.

4 Conclusion

This paper proposes using LSTM RNN models to predict order supply-demand gaps. LSTM RNN possesses the capability to process time series data. Key structural parameters including hidden layer node count, truncation length, and number of hidden layers significantly impact prediction accuracy. Through simulation experiments, we find the optimal truncation length is 2, meaning order information from 20 minutes before the prediction time slice has the greatest impact. Visual analysis of training data reveals substantial differences between

weekend and weekday order patterns, so we classify training data to improve accuracy. To demonstrate LSTM RNN' s superior prediction accuracy, we compare it with four other regression methods: BPNN, SVR, DTR, and RW. Experimental results based on RMSE comparisons show LSTM RNN achieves the smallest error, verifying its higher precision, better generalization capability, and stronger memory for historical data with varying intervals.

Future work will focus on two aspects: first, incorporating factors such as promotional activities and special holidays into the model, and adding bias to the loss function design based on practical considerations; second, improving model structure to account for real-time supply-demand fluctuations.

References

[1] Sainath T N, Weiss R J, Wilson K W. Multichannel signal processing with deep neural networks for automatic speech recognition [J]. IEEE//ACM Trans on Audio, Speech, and Language Processing, 2017, 25 (5): 965-979.

[2] Goodfellow I, Bengio Y, Courville A, Deep learning (adaptive computation and machine learning series) [M]. Cambridge: MIT Press, 2016.

[3] Silver D, Huang A, et al. Mastering the game of go with deep neural networks and tree search [J]. Nature, 2016, 529: 484-489.

[4] Yi H, Jung H J, Bae S. Deep neural networks for traffic flow prediction [C]// Proc of IEEE International Conference on Big Data and Smart Computing. 2017: 328-331.

[5] Mo Y J, Kim J, Kim J K, et al. Performance of deep learning computation with TensorFlow software library in GPU-capable multi-core computing platforms [C]// Proc of the 9th International Conference on Ubiquitous and Future Networks. 2017: 240-242.

[6] Ichinose A, Oguchi M, Takefusa A, et al. Evaluation of distributed processing of caffe framework using poor performance device [C]// Proc of IEEE International Conference on Big Data. 2016: 3980-3982.

[7] Abadi M, Agarwal A, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems [R]. Preliminary White Paper, 2015.

[8] Hinton G, Sabour S, Frosst N. Dynamic routing between capsule [C]// Proc of Conference on Neural Information Processing Systems. 2017.

[9] Wongsuphasawat K, Smilkov D, et al. Visualizing dataflow graphs of deep learning models in TensorFlow [J]. IEEE Trans on Visualization and Computer Graphics, 2018: 24 (1): 1-12.

[10] Gallardo N, et al. Autonomous decision making for a driver-less car [C]// Proc of the 12th System of Systems Engineering Conference. 2017.

[11] Gao Feng. Network traffic prediction based on neural network [C]// Proc of International Conference on Intelligent Transportation, Big Data and Smart City. 2015: 527-530.

[12] Chen Dawei. Research on traffic flow prediction in the big data environment based on the improved RBF neural network [J]. IEEE Trans on Industrial Informatics, 2017, 13 (4): 2000-2008.

[13] Highway Capacity Manual. Transportation research board, National Research Council, Washington, DC, 2000: 113.

[14] Karlaftis M, E. Vlahogianni. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights [J]. Transportation Research Part C: Emerging Technologies, 2011, 19 (3): 387-399.

[15] Levin M, Tsao Y D. On forecasting freeway occupancies and volumes (abridgment) [J]. Transportation Research Record, 1980, 773.

[16] Yu Haiyang, Wu Zhihai, Chen Dongwei, et al. Probabilistic prediction of bus headway using relevance vector machine regression [J]. IEEE Trans on Intelligent Transportation Systems, 2017, 18 (7): 1772-1781.

[17] Castro-Neto M, Jeong Y S, Jeong M K, et al. Onlinesvr for short-term traffic flow prediction under typical and atypical traffic conditions [J]. Expert systems with applications, 2009, 6 (3): 6164-6173.

[18] Chan K Y, Dillon T, Chang E, et al. Prediction of short-term traffic variables using intelligent swarm-based neural networks [J]. IEEE Transs on Control Systems Technology, 2013, 21 (1): 263-274.

[19] Vlahogianni E I, Karlaftis M G, Golias J C. Optimized and meta-optimized neural networks for short-term traffic flow prediction: a genetic approach [J]. Transportation Research Part C: Emerging Technologies, 2005, 13 (3): 211-234.

[20] Yu Xiaohan, Xiong Shengwu, Xiang Jianwen, et al. A campus traffic congestion detecting method based on BP neural network [C]// Proc of the 2nd International Symposium on Dependable Computing and Internet of Things. 2015.

[21] Chai Yanchong, Huang Darong, Zhao Ling. A short-term traffic flow prediction method based on wavelet analysis and neural network [C]// Proc of Chinese Control and Decision Conference. 2016: 7030-1034.

[22] Chen Dawei. Research on traffic flow prediction in the big data environment based on the improved RBF neural network [J]. IEEE Trans on Industrial Informatics, 2017, 13 (4): 2000-2008.

[23] Wang Peng, Zhao Gang, Yao Xingren. Applying back-propagation neural network to predict bus traffic [C]// Proc of the 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery. 2016.

[24] Yu Wanxia, Liu Lina; Zhang Weicun. [C]// Proc of the 8th International Conference on Intelligent Networks and Intelligent Systems. 2015: 141-144.

[25] Wang Dong, Cao Wei, Li Jian. DeepSD: supply-demand prediction for online car-hailing services using deep neural networks [C]// Proc of the 33rd International Conference on Data Engineering 2017: 243-254.

[26] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural Computation, 1997, 9 (8): 1735-1780.

[27] Ma Xiaolei, Ding Chuan, Luan Sen, et al. Prioritizing influential factors for freeway incident clearance time prediction using the gradient boosting decision trees method [J]. IEEE Trans on Intelligent Transportation Systems, 2017, 18 (9): 2303-2310.

[28] Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization [J/OL]. Eprint Arxiv, 2014. https://arxiv.org/abs/1409.2329v5.

[29] Schmidhuber J. Learning to control fast-weight memories: an alternative to dynamic recurrent networks [J]. Neural Computation, 1992: 4 (1): 131-139.

[30] Siegelmann H T, Sontag E D. Some recent results on computing with' neural nets' [C]// Proc of the 31st IEEE Conference on Decision and Control. 1992: 240-245.

[31] Fouladgar M, Parchami M, Elmasri R, et al. Scalable deep traffic flow neural networks for urban traffic congestion prediction [C]// Proc of International Joint Conference on Neural Networks. 2017: 2251-2258.

Note: Figure translations are in progress. See original paper for figures.