Journal of Computers Vol. 31 No. 3, 2020, pp. 195-205
doi:10.3966/199115992020063103015

Method for Predicting Travel Time of Motor Vehicle Based on Stack Noise Reduction Self-encoder

Rong Cheng[1*], Xuan Feng[1]

[1] Zhejiang Institute of Communications, Hangzhou, China
iongcheng@vip.sina.com

Received 18 May 2019; Revised 11 November 2019; Accepted 20 January 2020

Abstract. Estimating the travel time of any path (denoted by a sequence of GPS points) in a city is the key research problem of this work. It plays a vital role in traffic control, path planning, vehicle scheduling, and so on. Travel time prediction is a challenging proposition to predict the travel time of motor vehicles accurately. It is interfered with by many complex factors, such as Spatio-temporal correlation, traffic light influence, data sparse, etc. In most of the existing research, it estimates the travel time of motor vehicles in a single section or subpath, and it gets the result without considering the situation of intersections and traffic lights. It is difficult to predict accurately the travel time of a longer path. In this paper, the deep network based on stacked denoising autoencoder is proposed, which can directly estimate the travel time of any path and overcome the error accumulation problem of long path estimation. The experimental results of taxi trajectory data set show that the travel time prediction method proposed in this paper based on the stack noise reduction self-encoder has good prediction accuracy and algorithm stability.

Keywords: deep network, stack noise reduction self-encoder, taxi trajectory, travel time prediction

1 Introduction

With the continuous improvement of China’s economic development level, the urban population is increasing year by year and the urban scope is expanding, so that the urban population’s demand for road traffic is also increasing. Although the urban traffic infrastructure is also growing and improving, it still cannot meet the traffic demand brought by the development of urbanization. Excessive motor vehicles not only aggravate air pollution but also lead to frequent traffic jams. The increasingly severe congestion brings excellent pressure to the existing traffic facilities and seriously breaks up the demand for road traffic. It not only brings difficulties to traffic managers in traffic management and traffic policy formulation but also increases the travel cost and travel time of travelers.

As an essential indicator of the state of road traffic, travel time has received extensive attention. It can not only measure the service level of urban road facilities but also directly evaluate the travel plans of travelers. Travel time is also known as “vehicle travel time”. It indicates the time that the vehicle has a travel experience with a bright start and endpoint and a start and end time. By forecasting and analyzing travel time, traffic managers can measure the accessibility of traffic communities to assist in road planning and design. The forecast analysis of travel time can also be used to evaluate the service quality of roads and help traffic management. Because travel time is the most direct representation of travel plans, travelers tend to be more concerned with the travel time of specific trips than the details of travel paths [1]. When travelers search for candidate paths, accurate travel time estimates can help them plan paths better and avoid busy roads for alleviating traffic congestion. Now almost all electronic maps and online taxi services offer travel time estimates in their applications, such as Google Maps, Uber, and Didi Taxi. The quality of the assessment is critical to the user experience of these applications. Therefore, travel

* Corresponding Author

time prediction is an important issue and a hot research topic in path planning, navigation, and traffic dispatching. A large number of research results about travel time prediction have been published [2-4]. However, the travel time of the urban road network is affected by many factors, such as road conditions, weather, time, speed, and other factors. There are uncertain nonlinear relationships between travel time and these factors. The accuracy and applicable scope of existing prediction models are still limited. Overall, although there have been many research results and applications about travel time, complex traffic flow, and diverse traffic networks yet bring many challenges to the study of the travel time prediction of the motor vehicle. It's always the hotspot and difficulty to research how to predict the travel time of a given path accurately.

Accurate travel time prediction is essential for traffic managers and travelers. In highly urbanized areas, path-oriented travel time forecasts are more valuable to travelers. Travelers often expect to know travel times between multiple start and end points covering numerous road trips, rather than paying attention to traffic conditions on specific road sections. In this paper, we hope to study the travel time prediction problem in the urban traffic network with the support of massive vehicle trajectory data. The taxi trajectory data is a necessary data that intuitively reflects the traffic situation, with the advantages of the all-weather, full coverage of time and space. It not only records the vehicle's driving path but also directly represents the geometric characteristics of the urban road network. It makes it possible to collect any "origin-destination" (O-D) path. In this paper, the stacking noise reduction self-encoder is proposed that simulates the correlation between travel time and historical traffic trajectory. Firstly, the trajectory data is preprocessed and then sent to the stack noise reduction self-encoder to obtain the vehicle travel time prediction, model.

The main contributions of this paper are as follows:

(1) In order to estimate the travel time of a given trail accurately, this paper proposes a travel time prediction model based on a stacked denoising autoencoder. Focusing on estimating the travel times of individual road segments or sub-paths and then summing up these times, this model overcomes the disadvantage of low accuracy in the traditional travel time prediction method.

(2) A lot of experiments have been carried out on the taxi trajectory data of the local area of Second Ring in Chengdu. Compared with the deep network and GBDT algorithm, the experimental results show that the proposed travel time prediction algorithm based on stacked denoising autoencoder has good prediction accuracy and algorithm stability.

The rest of this paper is structured as follows. The second part briefly introduces the existing methods and some open problems to deal with travel time prediction of the motor vehicles. The third part is the introduction of the principle of stack noise reduction self-encoder and the construction process of travel time prediction of the motor vehicles, which mainly includes greedy layer-by-layer pre-training and deep network fine-tuning process. The fourth part is experimental verification and analysis. Taking the taxi trajectory data of the local area of Second Ring in Chengdu as an example, the proposed algorithm carries out travel time prediction analysis. It compares it with a deep network and GBDT algorithm by different objective functions. Finally, it summarizes and forecasts the next step in the paper.

2 The Related Work

In some research, it attempts to model travel-oriented travel time using complex traffic flow models. However, many problems still exist, such as lack of large amounts of raw data, inaccurate travel prediction for longer distances, and so on. For the availability of heavy vehicle historical trajectory data, it presents new opportunities for travel time prediction. For example, taxi trajectory data has been frequently used for predict travel time [5-6], bus trajectory data [7], mobile phone trajectory data [8], car navigation system data [9], etc. In theory, vehicle trajectory data can provide complete Spatio-temporal coverage of a particular path, and detailed information on departure points, termination points, trails, and travel times.

Facing the massive demand for travel time by travellers, service providers of advanced traffic management systems (ATMS) and advanced traveler information systems (ATIS) usually only publish average travel time for road segments, and assume that the travel time of the trip covering multiple sections is the sum of the travel times of the sections. The papers [10-11] focus solely on accurately estimating the travel time or speed of a single road segment, without considering the interrelationship between the roads. Travel time of a road is affected by many factors, such as the number of road intersections and traffic lights on a road. By merely adding the travel time of each section of the path, it does not get accurate results, especially during peak hours.

The above algorithms are based on the final summation of a single road segment evaluation, and the shortcomings are also evident. In some papers, it evaluates the travel time based on the entire path and overcomes the shortcomings of the individual assessment of the individual sub-segments. In the paper [12], it estimates the travel time of the modified road based on historical data for a given path. However, the historical mean-based model may lead to reduced accuracy, and the new query path may not be included in the historical data so that there may be data sparsity problems. In the paper [13], it constructs a landmark map based on the historical trajectory of the taxi. Each landmark on the map represents a road. It estimates the travel time distribution of the path according to the road map. However, since these landmarks are selected from the top k roads with the most massive traffic flow, those roads with a little traffic can't be accurately estimated. Besides, in the paper [14], it evaluates the travel time of the path based on the sub-tracks in the historical data by using the tensor decomposition to fill the missing data in the trajectory data. The algorithm effectively improves the prediction accuracy of the travel time. It still has the problem of sparse data, because there are many trajectories that few drivers can reproduce.

For urban road networks, the main challenge of the travel-oriented travel time prediction problem is the acquisition of traffic data with complete Spatio-temporal coverage and the construction of models covering all influence parameters. Next, we will discuss the research results that focus on the travel-time forecasting problem for the path. In the paper [15], it proposes a real-time travel-oriented travel time prediction model. The travel time of one trip is defined as the sum of the travel time of multiple road sections and the delay of the intersection. The travel time of a segment is dynamically calculated from the real-time and historical taxi trajectory data, and a similar trajectory estimates the intersection delay time. In their research results, it is limited to real-time or short-term forecasts, and the predicted results will change dynamically due to changes in traffic conditions in travel. The method proposed in the paper [16] uses time series prediction methods to predict travel-oriented travel time, and it considers the path as a more extended road segment without the details inside the road segment. This method can predict the future of any trip, but the performance is unstable, and the accuracy of the prediction is quite different in different departure periods. It suggests providing test efficiency and stabilizing performance by using high-performance prediction methods and adding more influencing factors. The external influencing factors, such as a particular time or meteorological conditions, have been shown to have an impact on travel time. In the paper [17], these external factors are used to predict travel-oriented travel time. Attempting to predict travel-oriented travel time, they used road segment attributes (speed limit requirements, road grades) and trip attributes (date, season, weather, etc.) to build a statistical model by the maximum likelihood method. Due to research area and climate constraints, it highlights the impact of winter snowfall on travel time. Since the coverage of the detect vehicle is limited, the applicability of the experimental results to the massive vehicle trajectory data needs further examination.

The interpretability of the tree-based integration model enables traffic decision-makers to understand the model's output better and to facilitate analysis of the influencing factors between traffic volumes. These characteristics make the tree-based integration method have a good application prospect in solving traffic problems. In the paper [18], random forest (RF) is applied to travel time prediction and performed best in IEEE ICDM competitions, which has distinct advantages over other algorithms. The gradient boosting decision tree (GBDT) has superior performance in prediction accuracy compared to traditional statistical methods and other integrated methods. In the paper [19], by applying the gradient lifting tree to travel time prediction, a multi-step advanced travel time prediction method based on the integrated algorithm is developed, and good results are obtained, especially when the traffic condition is suddenly changed. However, this method needs to do a troublesome preprocess of the input data firstly. The proposed method in this paper estimates the travel time of the whole path directly and overcome the error accumulation problem of long path estimation. Besides, our model is trained end to end, without any extra preprocessing.

3 Stack Noise Reduction Self-encoder

3.1 Self-encoder

Auto-encoder (AE) is an unsupervised neural network structure consisting of an input layer, a hidden layer, and an output layer. Its structure is shown in Fig. 1. In the hidden layer, it encodes the input data and then reconstructs the input data to obtain an abstract representation of the data by minimizing the reconstruction error.

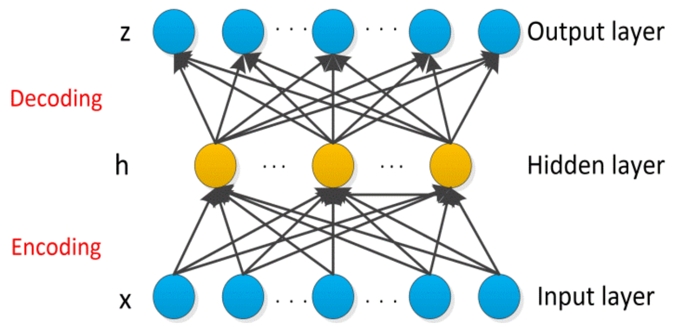

Fig. 1. Self-encoder structure

Let D_i and D_h represent respectively the number of cells from the input and hidden layers of the encoder. A set of untagged data sets X = $\{x_1, x_2, x_3, \ldots\}$ is Given, and $x_i \in \mathrm{R}^n$. In the encoding stage, the encoder f_θ converts the input vector x_i into a hidden layer representation h by a linear mapping and a nonlinear activation function:

$$h(x_i) = f_\theta(x_i) = f(W_1 x_i + b_1), \tag{1}$$

where $\theta = \{W, b\}$, $W_1 \in \mathrm{R}^{D_i \times D_h}$ is encoding weight matrix, $b_1 \in \mathrm{R}^{D_h}$ is offset vector.

In the decoding stage, the input vector x_i is reconstructed by a linear mapping of the hidden layer representation h, and the output vector z is obtained:

$$z_i = g_{\theta'}(h(x_i)) = W_2 h(x_i) + b_2, \tag{2}$$

where $\theta' = \{W_2, b_2\}$. $W_2 \in \mathrm{R}^{D_h \times D_i}$, $b_2 \in \mathrm{R}^{D_i}$ represents the decoding weight matrix and the offset vector, respectively.

From the perspective of model learning, it minimizes the reconstruction error of the input vector x_i and the output vector z by changing the relevant parameters, and it means the cost function of the input and output is minimized. For the given data set $\{(x^1, y^1), \ldots, (x^n, y^n)\}$, there is a self-encoder $x^n = y^n$, and its cost function is defined as:

$$J(W,b) = [\frac{1}{n}\sum_{i=1}^{n}(\frac{1}{2}\|z_i - y_i\|^2)] + \frac{\lambda}{2}\sum_{l=1}^{m_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(W_{ij}^{(l)})^2. \tag{3}$$

The first part of the above formula is the mean square error term, and the second part is the regularization term, whose purpose is to reduce the magnitude of the weight and prevent over-fitting. λ is the regularization coefficient.

The correlation network weight W and the offset value b can be obtained by minimizing the cost function $J(W, b)$; the process can be implemented by the Back-Propagation (BP) algorithm [20].

3.2 The Basic Principle of Stack Noise Reduction Self-encoder

The traditional self-encoder is equivalent to directly completing the direct copying of the input data or the output of the small change without any constraint. It will cause the model to generate a specific reconstruction error to some extent; there is insufficient adaptability to scenes where data distribution is inconsistent [21]. Compared to the automatic encoder, the noise reduction self-encoder trains the automatic encoder by adding a small amount of noise to the input layer to enable feature extraction that is more robust to the input data. As shown in Fig. 2, SDAE realizes the abstract feature representation of the original data by stacking multiple noise-reducing encoders to reconstruct the input. It means that it trains the first hidden layer through the input data and then takes the output of the first hidden layer as the first input of the second hidden layer until the desired data representation is completed.

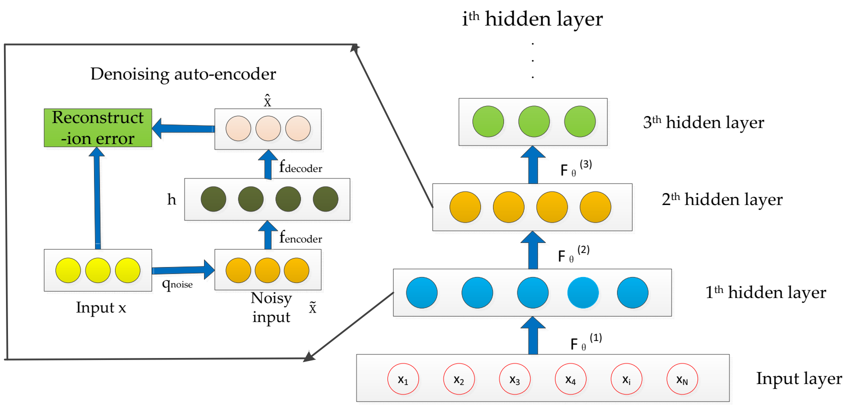

Fig. 2. The structure diagram stack noise reduction encoder

Add random noise to the original input through a random map: $\tilde{x} \rightarrow q(\tilde{x}|x)$, and map it to a hidden layer representation:

$$h = f_\theta(\tilde{x}_i) = f(W_1\tilde{x}_i + b_1) . \quad \textbf{(4)}$$

The activation function is described as:

$$f(a) = \max(0, a) . \quad \textbf{(5)}$$

And use the same way as the automatic encoder for input reconstruction:

$$z_i = g_{\theta^{'}}(h) = W_2 h + b_2 . \quad \textbf{(6)}$$

Similarly, the noise reduction encoder parameters $\{\theta,\ \theta^{'}\}$ are obtained by minimizing the cost function (Equation 4). Considering that SAE is a stack of multi-layer noise reduction encoders, W_{all} represents the weight matrix of a complete stack noise reduction self-encoding network, and b_{all} represents the offset matrix. They can be described as:

$$W_{all}, b_{all} = \underset{\theta_i,\ \theta_i^{'}}{\arg\min}\ J(W, b) . \quad \textbf{(7)}$$

Let $l \in \{1, \ldots, L\}$ represents the number of hidden layers of SDAEs, h_l represent the output vector of layer l, W_l represent the weights and b_l represent offsets of layer l. The feedforward DNN pre-trained with SAE can be described as:

For $l \in \{1, \ldots, L\text{-}1\}$,

$$h_{l+1} = f(W_{l+1}h_l + b_{l+1}) , \quad \textbf{(8)}$$

where $f(x)$ is the activate function for hidden layer and h_{l+1} is the final output characteristics.

3.3 Travel Time Prediction Model Construction

After completing the unsupervised greedy layer-by-layer pre-training, a fully connected layer is added at the top of the network, and the BP algorithm is used to complete the supervised fine-tuning of the whole model to complete the construction of the entire DNN model. In the entire DNN, for any $l \in \{1, \ldots, L\text{-}1\}$, the parameters of layer l are the same as the corresponding stack noise reduction encoder, and the topmost weight $\{W_L, b_L\}$ is randomly initialized, and has:

$$h_{L+1} = P(W_{L+1}h_L + b_{L+1}), \tag{9}$$

where P is the prediction function of DNN, and the fully connected layer is used here.

Finally, the BP algorithm is used to train the entire deep network.

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} L(F_\theta(C_i), T_i), \tag{10}$$

where $F(C) = P_{\theta_{L+1}}(f_{\theta_l}(\ldots f_{\theta_1}(\mathrm{C})))$ is the composite function of SDAEs, θ is the model parameters $\{W_l, b_l\}$, $l \in \{1, \ldots, L+1\}$, $L(\mathrm{X}, \mathrm{Y})$ represents the cost function of the entire network. The experiment will compare different cost functions. Different cost functions will be compared in the experiment.

The self-encoder of this experiment is a single hidden layer encoder that superimposes two encoders and connects with a fully connected layer to form a stack noise reduction self-encoder. Combined with the network's supervised training, the greedy layer-by-layer pre-training method successfully overcomes the training problem that the DNN model is easy to fall into the local optimal solution [22]. The complete DNN training process is shown in Fig. 3.

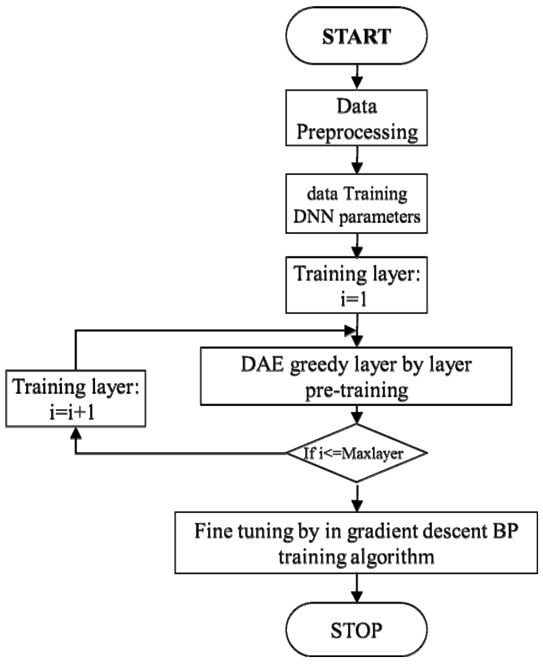
START
Data Preprocessing
data Training DNN parameters
Training layer: i=1
DAE greedy layer by layer pre-training
Training layer: i=i+1
If i<=Maxlayer
Fine tuning by in gradient descent BP training algorithm
STOP

Fig. 3. DNN training process

4 Experimental Verification and Analysis

4.1 Data Set Description

The experimental data set is from the local trajectory data of the Second Ring in Chengdu in November 2016. All trajectory data in the data range ([30.727818, 104.043333], [30.726490, 104.129076], [30.655191, 104.129591], [30.652828, 104.042102]) and the date range in (November 1, 2016) will be displayed with an accuracy of 2-4s. The order data will include the start and endpoint coordinates of the order and the start and end time.

This experiment takes 10052 trajectory data in the dataset. The most extended trajectory has 1042 GPS coordinates, and the shortest has 50 GPS coordinates. The longest travel time is 8655s, and the quickest is 151s. The track length distribution and travel time distribution are shown in Fig. 4 and Fig. 5.

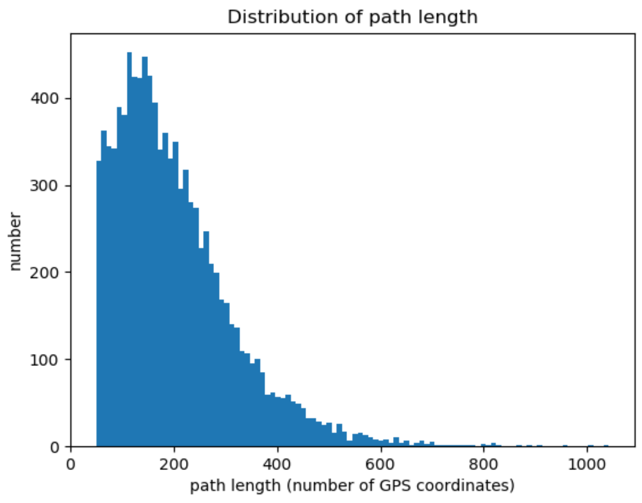

Fig. 4. The track length distribution

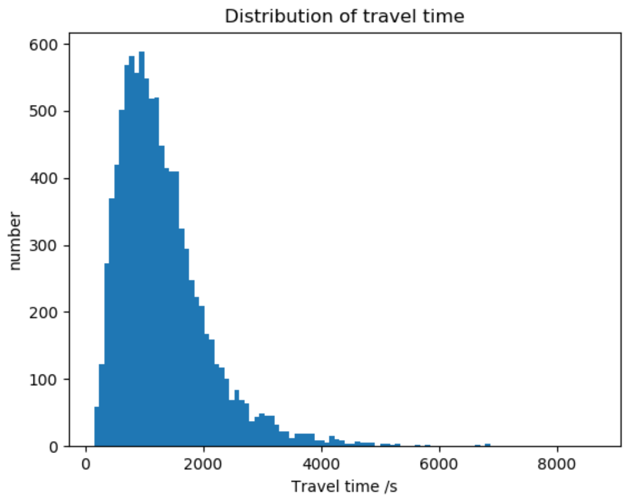

Fig. 5. The travel time distribution

4.2 Data Preprocessing

The GPS device records data once every fixed time interval. The original data may cause the model to be misled. For example, merely counting the number of GPS records can get travel time. To avoid this situation, it needs to pre-process the trajectory data in the experiment. The pre-processing process is to unify the number of GPS coordinate points of these tracks into the mode M of the number of GPS coordinate points of all trails. For the trajectory whose number of coordinate points is greater than M, the starting point and the ending point are guaranteed to be unchanged, and some coordinate points are randomly deleted to make the length consistent with M. For the trajectory whose number of coordinate points is less than M, the starting point and the ending point are guaranteed to be randomly interpolated, and the value is the average of the two GPS coordinate points before and after so that the length is consistent with M. Finally, standardize the entire data. The data characteristics of the experiment include the track start timestamp, the track longitude sequence, the track latitude sequence, the order start point longitude, the order start point latitude, the order endpoint longitude, and the order endpoint latitude. The data tag is the travel time to complete the order.

4.3 Performance Comparison

The experimental training model uses the verification set to control the number of training iterations and uses the grid search method to determine the best parameters of the model. After determining the optimal parameters, the test set was repeated to test 5 times and averaged. The deep learning framework PyTorch 0.4.1 is used in this experiment and implemented on Ubuntu 16.04LST (CPU: Intel (R) Core i5-6500T CPU@2.50GHz*4).

Among the following loss functions, *predicted$_i$* are predicted values, *truth$_i$* are true values, *truth$_i$* are regular coefficients, and *reg* are regular terms.

(1) Average squared difference loss (Loss$_{\mathrm{MSE}}$)

$$Loss_{MSE} = \frac{1}{n}\sum_{i}^{n}(predicted_i - truth_i)^2 + \lambda \times reg\,. \tag{11}$$

(2) Average absolute error loss (Loss$_{\mathrm{MAE}}$)

$$Loss_{MAE} = \frac{1}{n}\sum_{i}^{n}\left|predicted_i - truth_i\right| + \lambda \times reg\,. \tag{12}$$

(3) Average relative error loss (Loss$_{\mathrm{MAPE}}$)

$$Loss_{MAPE} = \frac{1}{n}\sum_{i}^{n}\frac{\left|predicted_i - truth_i\right|}{truth_i} + \lambda \times reg\,. \tag{13}$$

(4) Symmetric average relative error loss (Loss$_{\mathrm{SMAPE}}$)

$$Loss_{SMAPE} = \frac{1}{n}\sum_{i}^{n}\frac{\left|predicted_i - truth_i\right|}{(predicted_i + truth_i)/2} + \lambda \times reg\,. \tag{14}$$

Through the grid search method, the model parameters of different loss functions are selected and set as Table 1.

Table 1. Model parameter selection table for different loss functions

	Loss$_{\mathrm{MSE}}$	Loss$_{\mathrm{MAE}}$	Loss$_{\mathrm{MAPE}}$	Loss$_{\mathrm{SMAPE}}$
Hidden_dim1	256	64	256	128
Hidden_dim2	16	32	128	64
Batch size: Mini_batch	128	128	128	128
Regular term coefficient: λ	1e-3	1e-3	0	1e-4
Noise parameters: scale	1e-4	1e-4	1e-3	1e-3

In Table 1, Hidden_dim1 means the number of hidden layer nodes of the first self-encoder and Hidden_dim2 means the number of hidden layer nodes of the second self-encoder.

The experimental results are as Table 2.

Table 2. Forecast results indicators of different loss function models

	Loss$_{\mathrm{MSE}}$	Loss$_{\mathrm{MAE}}$	Loss$_{\mathrm{MAPE}}$	Loss$_{\mathrm{SMAPE}}$
MAE	328.08	**317.79**	328.03	319.27
RMSE	**479.04**	490.49	526.58	489.43
MAPE (%)	28.47	25.09	**23.54**	24.79

Result analysis. Firstly, statistically analyze the travel time in the data set whose distribution is shown in Fig. 5. The delivery is overall to the left. The data whose travel time less than 1658s occupies 75% of the entire data set. The data set holds a part of the outliers with a high score. LossMSE is a widespread loss function in regression problems. It can be seen from the experimental results that the RMSE index is excellent, MAE, and MAPE are not good. Because LossMSE with MAE and MAPE is less robust to outliers. It gives higher weight to the outliers at the expense of the prediction of other standard data points, thus reducing the overall performance of the model. One advantage over LossMSE is that the former is less sensitive and more inclusive of outliers. LossMAE calculates the absolute value of the error, there is no square term, and the penalty is the same. Its RMSE index is not as good as LossMSE, but the overall performance of the model is better. RMSE indicator of LossMAPE is the worst of several comparison experiments. Because the pursuit of the smallest MAPE lets the model can fit better to lower real values and deviate from those larger actual values, the overall MAPE is reduced while the RMSE is greatly increased. To overcome the defect of LossMAPE, LossSMAPE changes the denominator of LossMAPE to the average value of predicted value and real value. Using this loss function can effectively prevent the influence of outliers on the overall score. When the travel time reaches a significant amount, the denominator term can relatively weaken the effect. It's seen from the experimental results that LossSMAPE is not excellent on each independent index in several independent experiments, but the overall performance of the model is the best.

In this experiment, three models were used for performance comparison, namely stack denoising self-encoder (SDAE), deep neural network (DNN), and gradient lifting decision tree (GBDT) [23]. The SDAE used in this experiment is based on the loss function of SMAPE, and its overall performance is the best in the experiment, which can be seen in 4.3.1. The DNN network model is consistent with the depth of the stack noise reduction self-encoder that takes SMAPE as the loss function, and the other super parameters are also the same. GBDT is consistent with the input of the above experiment. GBDT_log is a logarithmic transformation of the travel time of Fig. 5, which makes it close to the normal distribution. The data distribution is shown in Fig. 6.

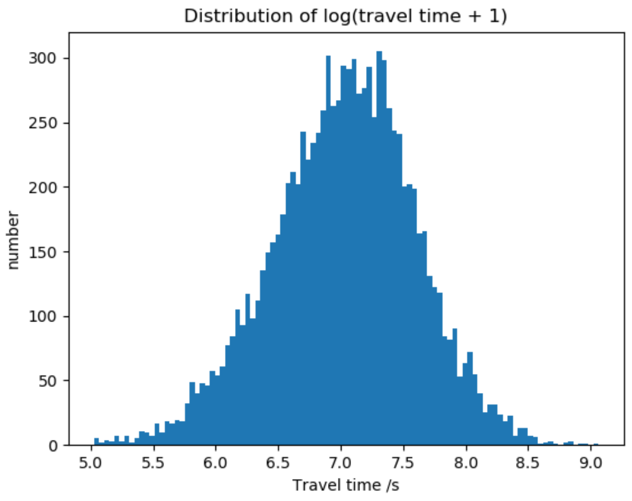

Fig. 6. Travel time distribution after logarithmic transformation

The experimental results are as Table 3.

Table 3. Different algorithm prediction results

	SDAE	DNN	GBDT	GBDT_log
MAE	319.27	326.97	288.97	**284.88**
RMSE	489.43	501.58	**425.55**	432.89
MAPE (%)	**24.79**	29.61	27.66	25.62

Result analysis. From the experimental results, it can be seen that SDAE has better performance than DNN and exceeds DNN in all three indicators. Otherwise, SDAE's MAPE indicator reached the best of the above four experiments. The overall performance of SDAE and DNN is more inferior to the GBDT model. One possible reason is that the number of training samples is not large enough to play the advantage of the neural network. The travel time of the training samples is logarithmically transformed to reduce the outliers that are too large or too small so that the distribution is closer to the normal distribution, which is beneficial to the training of the model. From the results of GBDT and GBDT_log, it can be seen that the former's RMSE is slightly smaller than the latter, and the other two indicators are not right. It shows that GBDT_log has a good fit in most travel time, not biased by a few significant outliers, and more robust to outliers than GBDT.

In general, SDAE overcomes the local optimal solution of the DDN model so that the overall performance is better than DNN. Compared with GBDT and GBDT_log, SDAE does not need complicated and troublesome preprocessing of the input data, and the experimental results show that they have advantages and disadvantages in parameter comparison. It could be predicted that SDAE may have better performance when it takes advantage of the neural network in a larger training sample size.

5 Conclusion

In this paper, we propose an end-to-end framework based on deep neural networks to estimate the travel time for any given trail. Our model can effectively capture the spatial and temporal dependencies in the given trail at the same time. It shows good prediction accuracy and algorithm stability on a large scale real-world taxi trajectory dataset. However, our model does not consider various factors that may affect travel time, such as driving habits, weather conditions, road speed limits, etc. In the next work, we will consider these factors to make the prediction model more robust and accurate.

Acknowledgements

This paper is supported by the Scientific Research Project of the Zhejiang Provincial Communication Department (2018047).

References

[1] T. Xu, X. Li, C. Claramunt, Trip-oriented travel time prediction (TOTTP) with historical vehicle trajectories, Frontiers of Earth Science 12(2)(2018) 253-263.

[2] R. Long Cheu, C. Xie, D.H. Lee, Probe vehicle population and sample size for arterial speed estimation, Computer- Aided Civil and Infrastructure Engineering 17(1)(2002) 53-60.

[3] A. Hofleitner, R. Herring, A. Bayen, Arterial travel time forecast with streaming data: a hybrid approach of flow modeling and machine learning, Transportation Research Part B: Methodological 46(9)(2012) 1097-1122.

[4] X. Zhan, S. Hasan, S.V. Ukkusuri, C. Kamga, Urban link travel time estimation using large-scale taxi data with partial information, Transportation Research Part C: Emerging Technologies 33(4)(2013) 37-49.

[5] E. Jenelius, H.N. Koutsopoulos, Travel time estimation for urban road networks using low frequency probe vehicle data, Transportation Research Part B: Methodological 53(4)(2013) 64-81.

[6] L. Tang, Z. Kan, X. Zhang, X. Yang, F. Huang, Q. Li, Travel time estimation at intersections based on low-frequency spatial-temporal GPS trajectory big data, The American Cartographer 43(5)(2016) 417-426.

[7] T.K. Sellis, N. Roussopoulos, C. Faloutsos, The R+-Tree: A Dynamic Index for Multi-Dimensional Objects, in: Proc. 13th International Conference on Very Large Data Bases, 1987.

[8] H. Bar-Gera, Evaluation of a cellular phone-based system for measurements of traffic speeds and travel times: a case study from Israel, Transportation Research Part C: Emerging Technologies 15(6)(2007) 380-391.

[9] M. Jones, Y. Geng, D. Nikovski, T. Hirata, Predicting Link Travel Times from Floating Car, in: Proc. 16th International IEEE Conference on Intelligent Transportation Systems, 2013.

[10] R. Sevlian, R. Rajagopal, Travel time estimation using floating car data. <https://arxiv.org/abs/1012.4249>, 2010.

[11] B. Pan, U. Demiryurek, C. Shahabi, Utilizing real-world transportation data for accurate traffic prediction, in: Proc. IEEE 12th International Conference in Data Mining, 2012.

[12] M. Rahmani, E. Jenelius, H.N. Koutsopoulos, Path travel time estimation using low-frequency floating car data, in: Proc. 16th International IEEE Conference in Intelligent Transportation Systems, 2013.

[13] J. Yuan, Y. Zheng, X. Xie, G. Sun, T-drive: enhancing driving directions with taxi drivers' intelligence, IEEE Transactions on Knowledge and Data Engineering 25(1)(2011) 220-232.

[14] Y. Wang, Y. Zheng, Y. Xue, Travel time estimation of a path using sparse trajectories, in: Proc. 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014.

[15] W.H. Lee, S.S. Tseng, S.H. Tsai, A knowledge based real-time travel time prediction system for urban network, Expert Systems with Applications 36(3)(2009) 4239-4247.

[16] Y. Jiang, X. Li, Travel time prediction based on historical trajectory data, Annals of GIS 19(1)(2013) 27-35.

[17] E. Jenelius, H.N. Koutsopoulos, Travel time estimation for urban road networks using low frequency probe vehicle data, Transportation Research Part B: Methodological 53(4)(2013) 64-81.

[18] B. Hamner, Predicting travel times with context-dependent random forests by modeling local and aggregate traffic flow, in: Proc. 2010 IEEE International Conference on Data Mining Workshops, 2010.

[19] Y. Zhang, A. Haghani, A gradient boosting method to improve travel time prediction, Transportation Research Part C: Emerging Technologies 58(1)(2015) 308-324.

[20] H.C. Shin, M.R. Orton, D.J. Collins, S.J. Doran, M.O. Leach, Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data, IEEE Transactions on Pattern Analysis and Machine Intelligence 35(8)(2012) 1930-1943.

[21] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, X. Chen, A sparse auto-encoder-based deep neural network approach for induction motor faults classification, Measurement 100(89)(2016) 171-178.

[22] G.E. Hinton, S. Osindero, Y.W. Teh, A Fast Learning Algorithm for Deep Belief Nets, Neural Computation 18(7)(2006) 1527-1554.

[23] J. Friedman, T. Hastie, R. Tibshirani, The Elements of Statistical Learning, Springer, Berlin, 2009.