Neural Computing and Applications
https://doi.org/10.1007/s00521-021-06560-z

ORIGINAL ARTICLE

GSTA: gated spatial–temporal attention approach for travel time prediction

Alkilane Khaled[1] · Alfateh M. Tag Elsir[1] · Yanming Shen[1]

Received: 4 May 2021 / Accepted: 15 September 2021

Abstract
Accurate travel time prediction between two locations is one of the most substantial services in transport. In travel time prediction, origin–destination (OD) method is more challenging since it has no intermediate trajectory points. This paper puts forward a deep learning-based model, called Gated Spatial–Temporal Attention (GSTA), to optimize the OD travel time prediction. While many trip features are available, their relations and particular contributions to the output are usually unknown. To give our model the flexibility to select the most relevant features, we develop a feature selection module with an integration unit and a gating mechanism to pass or suppress the trip feature based on its contribution. To capture spatial–temporal dependencies and correlations in the short and long term, we propose a new pair-wise attention mechanism with spatial inference and temporal reasoning. In addition, we adapt and integrate multi-head attention to improve model performance in case of sophisticated dependencies in long term. Extensive experiments on two large taxi datasets in New York City, USA, and Chengdu, China demonstrate the superiority of our model in comparison with other models.

Keywords Deep learning · Machine learning · Travel time prediction · Spatial–temporal attention

1 Introduction

Travel time prediction is regarded as one of the most critical services for travelers and vehicle operators. It effectively helps to mitigate traffic congestion as it can be utilized in different kinds of applications such as location-based applications, stochastic routing, ride sharing, navigation applications, trip planning, and traffic monitoring [26]. Given the vitality and crucial need for travel time prediction, travel time prediction has caught many researchers' attention [2, 13, 31, 35]. Travel time prediction methods can fall into three types: segment-based, path-based, and origin–destination based [34]. In segment-based approaches, GPS data are utilized to calculate travel times for each road segment, then the travel time for the whole road is the sum over all segments. Alternatively, path-based approaches depend on predicting the time for sub-paths taking into account the time spent on the crossroads [10]. In contrast, origin–destination (OD) based approaches ignore trajectory information while calculating the travel time. While the first two approaches face various challenges such as route mapping and data sparsity, the OD-based approaches overcome the aforementioned challenges by avoiding the expensive computations that come from finding the route first and then compute the travel time [1].

Many researchers have proposed solutions for travel time prediction, especially in the OD-based domain. Statistical methods in [4, 14, 30] were proposed to predict short-term travel time based on traffic statistical analysis and historical trips. However, these statistical models are not appropriate when the data includes nonlinear feature interactions, where they are unable to capture sophisticated spatial–temporal correlations. Machine learning-based approaches such as [17, 19, 24, 34] were also proposed to overcome the weaknesses in statistical methods and provide more accurate predictions. Recently, many deep learning models [35, 37, 40] have been developed.

✉ Yanming Shen
shen@dlut.edu.cn

Alkilane Khaled
khaledalkilane@mail.dlut.edu.cn

Alfateh M. Tag Elsir
m.fateh@mail.dlut.edu.cn

[1] School of Computer Science and Technology, Dalian University of Technology, Dalian, China

Published online: 27 September 2021

Abundant works in this domain such as [18, 33] have used RNN, LSTM, and CNN to catch the spatial–temporal dependencies between trips. Some of them merged RNN with CNN as in [33, 37], while others in [23, 38] employed historical trajectories and road networks to come up with better models. Some researches have applied the attention mechanism as in [29], while others have applied graph neural network as in [13]. Nonetheless, previous studies still have some drawbacks since RNN experiences difficulties related to the long training period and the slow inference speed, as it doesn't support parallel computing. Also, ignoring the vanishing and exploding gradient problem, that happens due to recurrence and back-propagation with so many layers in deep models, makes this kind of researches less efficient. Besides, some of these models failed in capturing complex and nonlinear spatial–temporal correlations.

Previous studies usually experienced one or more of the following limitations. The most common problem with travel time prediction is data sparsity. It is quite common in reality that some locations in the city either don't have any historical travel records or don't have enough travel records at different time intervals of the day. Most previous studies neglected to tackle this problem; therefore, their models' performance is questionable when it comes to real scenarios. Furthermore, the type, volume, and diversity of the used data are important factors when verifying the performance of the proposed method in different environments and traffic conditions. Previous studies have used either datasets that concentrate on private environments (freeways, highways,... etc.) or a single dataset, which is insufficient to prove the efficiency of the method under different conditions. Since traffic varies frequently based on weather conditions, leading to a large variation in travel time, weather data is another significant factor to be considered. Several previous research doesn't employ weather data in their methods, which greatly affects their methods' performance in abnormal weather. Besides the data limitations, some of the former research is based on the assumption of estimating the travel time of separate road segments and then aggregating those travel times. By doing so, it is hard to properly characterize complicated factors such as time spent in intersections and waiting for traffic lights, and might result in large accumulated errors, lowering the overall estimation accuracy.

Moreover, several previous methods have considered one of the spatial and temporal factors while ignoring the other one, i.e. methods that only rely on RNN (LSTM or GRU) are unable to model the spatial correlations which are very important in travel time prediction problem. Even though some of them have integrated RNNs with CNNs in a simple way using ConvLSTM, this simple integration is not capable of modeling sophisticated traffic data. Besides, long training time is another issue for these methods since RNN doesn't support parallel computing.

To overcome the above-mentioned limitations, we propose a deep learning-based approach to predict travel time. In this approach, we design a new jointly pair-wise spatial–temporal attention mechanism to capture spatio-temporal correlations and dependencies. We propose a feature selection module aims to select the most relevant features using two gating mechanisms (GRN and GLU). We solve the data sparsity issue by using k-means and geo-hashing algorithms for trip clustering. We use two real-world large-scale datasets from two different cities to verify the efficiency of the proposed method's performance under different environments and traffic conditions. Additionally, we employ weather data while conducting extensive experiments under extreme weather conditions to verify the effectiveness of the proposed approach under such situations. This approach is OD-based and predicts the travel time for the entire path. In this way, we can capture the time spent at all the intersections along the route, and the waiting time at traffic lights, as well as avoid large accumulated errors. The results from two large-scale datasets indicate that we can produce accurate travel time predictions.

The main contributions of this paper are as follows:

- We apply the K-means algorithm with geo-hashing technique to divide each city into different clusters and small grids (200×200 m). Therefore, the model can easily find neighboring trips located in the same cluster or grid in case of spatial or temporal sparseness, where the location is not covered by any records or covered by a few records, which would not be adequate to predict the travel time that fluctuates based on various status (e.g., peak period, weekends,... etc).
- While numerous trip features are available, their correlation and contribution to the target are usually anonymous. We design a feature selection module which serves as a linkage module. This module aims to elicit nonlinear patterns and correlations among various input features and select the most relevant ones based on their contributions to the output. To do so, we employ a combination of gated residual network and gated linear unit. This combination provides the model with the flexibility to choose the maximum applicable factors and do nonlinear processing as and when needed.
- We propose a new jointly pair-wise attention mechanism to capture nonlinear and complicated spatial–temporal dependencies and correlations using multi-head attention to parallelize the work and reduce the computation time. Also, we utilize residual connections through some model parts to regulate identification

mapping and avoid exploding and vanishing gradient problems.
- We conduct extensive experiments on two real-world large-scale datasets from two different cities (NYC and Chengdu). We also examine our method under abnormal and extreme weather conditions. The results demonstrate the efficiency and effectiveness of our method compared to other methods.

The rest of the paper is organized as follows: Sect. 2 reports the literature of previous researches on travel time prediction. The problem statement is defined in Sect. 3. In Sect. 4, the methodology and the architecture of our proposed model with all of its components are explained in detail. Section 5 presents a discussion on our experimental results. Lastly, a brief conclusion to this paper is summarized in Sect. 6.

2 Related work

Previous works in travel time prediction have applied three types of models: statistical-based, machine learning-based, and deep learning-based. Representative statistical models are Auto-regressive Integrated Moving Average (ARIMA) [4] and Naïve Bayes [14]. ARIMA is adopted in [4] to predict short-term freeway traffic data. In paper [30], the proposed model considered the predicted freeway travel time as the median of travel times for historical trips. In past, people attempted to predict the short-term travel time depending on a statistical analysis of traffic. These traffic information were sequential values. Many other pieces of research used additional input, like entry data from highway detectors and historical trips to improve their forecasting accuracy. Even so, since the data includes lots of nonlinear features that are sophisticated, it was hard for these models to catch the linked invisible nonlinear paradigm. These models are not able to fully register and recognize the sophisticated spatial–temporal paradigm within the massive traffic data. Machine learning (ML) based models are proposed to overcome the previous challenges. Myung et al. [24] presented a technique that depends on K-Nearest Neighbor (KNN) to forecast the travel time. Results have shown that KNN gives a promising structure for assembling travel time records from various periods with comparable possible traffic. Many other methods such as support vector regression (SVR) [17] and XGBoost [19] have shown their effectiveness in forecasting travel time as well. Nevertheless, ML-based methods are mostly focused on the temporal part of the data and ignore the spatial dependencies.

Recently, deep learning impressively outperforms statistical based and ML-based methods. It extracts the spatial–temporal correlations simultaneously from big data. In [37], a multi-task learning model (MTLM) for travel time estimation was presented, which recommends the suitable transportation mode for users, and then forecasts the linked travel time of the trajectory. While LSTM was used to capture the short and long temporal dependencies in paper [39], many works combined RNN with CNN to come up with a better model as in paper [15]. With this combination, they tried to avoid the weakness of using LSTM alone which comes from the fact that LSTM is incapable to model the spatial correlations which are very important for forecasting the travel time problem. Also, RNNs are limited to process consecutive data one after one, which restricts the parallelism computations. In paper [38], authors first matched the OD locations with their affiliated (historical) trajectories to road networks, and employ road segment embedding to characterize their spatial attributes, then corresponded to the time-stamps linked with trajectories to time slots and used time slot embedding to perform the temporal attributes.

Lately, some works take the advantage of combining ML with a deep neural network to achieve ensemble models. TTE-ensemble model was proposed in [40], used the gradient boosting decision tree (GBDT) to process the low dimensional features and select the DNN to treat high dimensional ones then merged GBDT and the DNN. Likewise, [31] presented an ensemble method based on a hybrid model that merged the gated recurrent unit (GRU) with XGBoost. In [36], authors proposed a method to forecast segment travel times considering the connections and dependencies among links travel times. They also investigated the possibility of forecasting the segment travel time mean and reliability indicators by utilizing the nonlinear autoregressive with exogenous inputs (NARX) method and feedforward neural network. A combination method called LSTM-CNN was suggested in [12] to estimate the highway travel time by combining LSTM and CNN with the attention mechanism and the residual network. By factoring in the traffic flow and the respective placement of automated number plate recognition (ANPR), the authors were able to partition the roadway into numerous parts. ANPR is used to determine the average travel time for every section per period, which is then passed to LSTM to estimate the journey time.

Recent works motivated by the success of the attention mechanism in natural language processing applied this mechanism to the problem of travel time prediction. Fang et al. [13] presented a Contextual Spatial-Temporal Graph Attention Network called ConSTGAT. In this model, a spatial–temporal graph neural network to achieve attention mechanism was proposed, then a computationally effective model employed convolutions and multi-task learning to get better results. In [29] an efficient model depends on a

feed-forward network (FFN) with Multi-factor self-Attention (FMA-ETA) was designed. The multi-factor self-attention mechanism is used to deal with various category features. Although FMA-ETA uses this mechanism to learn trip correlations, it confronts some obstacles while trying to comprehend complex nonlinear spatial dependencies and it ignores temporal dependencies between trips. Also, it results in less influence as it includes recurrent feedforward which greatly affects training time while may lead to the vanishing and exploding gradient problem.

Many researchers have considered the feature selection as an important method that helps the model to give more accurate results. In [25], authors suggested a unique deep network approach for learning characteristics across multiple modalities. They provided a group of tasks for multimodal learning and describe how to train deep networks that learn features to overcome these tasks. The authors of [3] developed traffic classifiers based on dynamically derived characteristics, representing the complicated paradigms extracted from the multidimensional nature of traffic and indirectly conveying information in a "multi-modal" manner. A combination of CNN and GRU were proposed in [21, 22] to extract latent features and identify numerous relationships across virtual machines based on historical occupations.

Table 1 illustrates the different categories of reviewed travel time prediction methods based on their main distinctive characteristics. From this table, we can notice that each method can be classified into one of the following categories: OD-based, segment-based, and path-based based on the data type used. Also, methods can be categorized as statistical, traditional machine learning, or deep learning based on the architecture of the model. Additional information, such as the spatial and temporal modeling types were also provided in this table.

3 Problem formulation

In this section, we give preliminaries and establish the main representation of the problem deliberated in this paper. We focus on processing OD trip data rather than anticipate the middle trajectory data. The formula expression of OD taxi trip is presented as follows:

Definition 1 (*OD Trip*): OD trip is represented by TR (PU_{loc}; DO_{loc}; PU_{dt}; DO_{dt}; E, T).

Where PU_{loc} denotes the pick-up location and DO_{loc} denotes the drop-off location. Both pick-up and drop-off locations are GPS coordinates with latitude and longitude points. Due to the difference in traffic volume at peak and off-peak hours throughout the day with the difference between weekdays and weekends, we denote the datetime for pick-up as PU_{dt} and drop-off as DO_{dt} for each trip in the dataset. We use $E = \{E(i) - -i = 1, 2, \ldots, m\}$ to define the set of remaining extracted features as distance, speed, weather data, ...etc. where m is the number of features. T denotes the trip duration for trip TR. We calculate trip duration which is considered as ground truth of our model as the difference between drop-off and pick-up time with the following: $T = DO_{dt} - PU_{dt}$.

Table 1 Categorization of existing travel time prediction methods based on their characteristics

Method	Data type	Model type	Spatio–temporal Modeling type	Consider Road network
ARIMA [4]	Path-based	Statistical	–	×
DLM [14]	Segment-based	Statistical	Adaptive DLM	×
SVR [17]	OD-based	Machine learning	–	×
KNN [24]	Path-based	Machine learning	–	✓
XGB-IN [19]	OD-based	Machine learning	–	×
MTLM [37]	Segment-based	Deep learning	CNN, ResNet	✓
DEEPTRAVEL [39]	Segment-based	Deep learning	BiLSTM	✓
ConSTGAT [13]	Path-based	Deep learning	GNN, attention	✓
DeepTTE [33]	Segment-based	Deep learning	LSTM, attention	×
TTE-Ensemble [40]	OD-based	Deep learning	DNN, GBDT	×
RSTN [15]	Path-based	Deep learning	Conv-LSTM	✓
FMA-ETA [29]	OD-based	Deep learning	FFN, Self-attention	×
Hybrid model [31]	Path-based	Deep learning	GRU, XGBoost	×
DeepOD [38]	OD-based	Deep learning	GNN	✓

Definition 2 (*Query*): trip query is represented by Q_{TR} (PU_{loc}; DO_{loc}; PU_{dt}) which takes a pick-up and drop-off location with the datetime for pick-up.

Definition 3 (*Travel Time prediction*): travel time prediction problem is defined as follows: Given a query Q_{TR}, using historical trips records in dataset Π = {TR(i) |i=1, 2,..., N}, where N is the number of trips in dataset, our goal is to predict the trip duration $\widetilde{T}$.

4 Model architecture

Figure 1 illustrates the structure of our model. The first part of this figure is the data pre-processing part, where we divide the data into spatial, temporal, weather, and additional extracted data. We apply entity embedding, cyclic encoding, and normalization to these parts, respectively. The second part of the figure is the spatial–temporal attention block with a feature selection module. The last part is the multi-head attention block with the output layer. In this section, we describe each part in detail.

4.1 Data pre-processing

We used two real-world large-scale datasets which are publicly available with millions of records that contain trip information such as origin and destination locations as a pair of pick-up and drop-off GPS coordinates, date and time for pick-up and drop-off, trip duration, and many other features. To prove the efficiency of our model in different traffic conditions and patterns, we validated our model using data from two major cities in two different countries. The first dataset is "NYC Yellow Taxi Trip" dataset that contains taxi trips from January to December, 2016 in New York City, USA [7]. The second dataset is "Didi Taxi Trip" dataset that contains taxi trips in September and October, 2018 in Chengdu, China [8]. Table 2 illustrates the statistics of both datasets. From this table, we can see that the NYC dataset has a total of 130 million trips with an average of 356,164 trips each day. The average trip duration is 15.36 min and the average trip distance and speed are 3.195 km and 17.480 km/h, respectively. On the other hand, the Chengdu dataset has a total of 8 million trips with an average of 131,147 trips each day. The average trip duration is 13.53 min and the average trip distance and speed are 5.137 km and 24.95 km/h, respectively. Since traffic varies a lot at different time intervals of the day (e.g., peak and off-peak periods), and between weekdays, weekends, and holidays, the dataset which is used to train and test the prediction model

Table 2 Statistics of NYC and Chengdu taxi trips datasets

	NYC	Chengdu
# total trips	130×10^6	8×10^6
# of trips per day	356,164	131,147
Avg trip time (min)	15.36	13.53
Avg trip distance (km)	3.195	5.137
Avg trip speed (kmh)	17.48	24.95
# of trips in peak	28.773×10^6	1.856×10^6
# of trips in off-peak	101.227×10^6	6.144×10^6
Avg peak time (min)	19.37	16.07
Avg off-peak time (min)	11.58	9.43
# of trips in weekdays	91.99×10^6	5.473×10^6
# of trips in weekends	33.46×10^6	1.62×10^6
# of trips in holidays	4.55×10^6	0.907×10^6

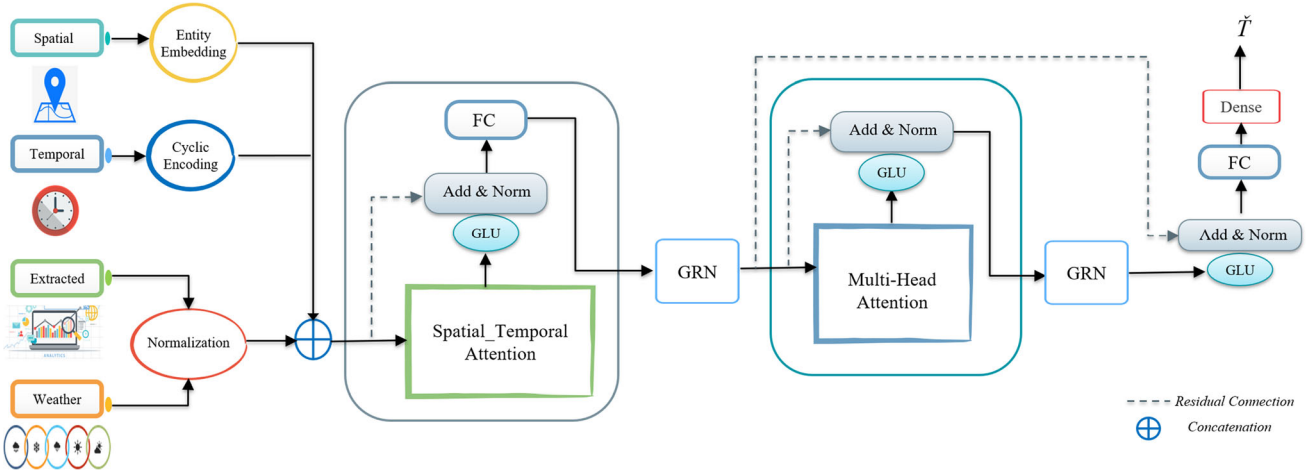


Fig. 1 GSTA architecture. We divide the data into: spatial, temporal, extracted, and weather data. All parts are grouped together after pre-processing. STA, MHA, GRN, GLU, and residual connections are then employed to compute the final prediction

should reflect this variation in traffic conditions. Table 2 shows that the two datasets we used involve a large number of trips in each pattern of different traffic conditions. The NYC dataset has a total of 28.773 and 101.227 million trips with an average trip duration of 19.37 and 11.58 min in peak and off-peak periods, respectively. It also has a total of 91.99, 33.46, and 4.55 million trips on weekdays, weekends, and holidays, respectively. The Chengdu dataset has a total of 1.856, and 6.144 million trips with an average trip duration of 16.07, and 9.43 min in peak and off-peak periods, respectively. It also has a total of 5.473, 1.62, and 0.907 million trips on weekdays, weekends, and holidays, respectively.

4.1.1 Outliers

After investigating the datasets, we filtered out all anonymous trips which didn't have either the pick-up or drop-off location coordination. Then, we dropped all trips out of the city border. Moreover, we removed all unbelievably trips with unreal information such as trips with a duration smaller than 3 min or larger than 5 h, and trips with a distance smaller than 500 m or bigger than 100 Km. Where the speed limit is almost $\sim$60/KMH in urban cities, trips with speed exceeding this limitation were omitted. Finally, we removed trips with outlier locations like trips with pick-up or drop-off inside the water for the NYC dataset.

4.1.2 Feature extraction

To increase the travel time prediction accuracy, we extracted some additional features that will help to represent the trip. *Speed* is a very important feature to calculate trip duration, so we added approximate speed to each trip by dividing the trip distance by its duration. Due to the time distribution in the datasets, we added two new features which are *peak* and *off-peak* hours since the traffic congestion has a high impact on travel time, and to differentiate between trips that happened in peak periods where the traffic is high and others in off-peak periods where traffic is low. After analyzing the datasets, we chose the peak hours between 7:00 $\sim$ 10:00 am as the morning peak and 17:00 $\sim$ 20:00 as the evening peak.

To enrich the spatial data and avoid data sparsity, we divided each city into small grids with the same size $(N * N)$ using geo-hashing techniques and mapped each trip to its related *geo-hash grid*, if the grid size N is large then the model error will increase. We have trained and tested our proposed method on both datasets (NYC and Chengdu) with range values of $N = [50, 100, 200, 500, 700]$, the best prediction results we got with the smallest MAE and MAPE when N is 200 m for both datasets. Also, we divided each city into clusters using the K-Means method. Through the clustering step, the parameter K might be overestimated or underestimated. A large value of K will drive to an overstated raise of feature dimension, which makes training the model more difficult. A small value of K will impact the model accuracy. We have also trained and tested our proposed method on both datasets with range values of $K = [25, 50, 100, 150, 200]$. The best prediction results we got with the smallest MAE and MAPE when K is 100 for both datasets. We added *cluster number* to each related trip and provided *cluster-center* for each cluster mapped to each trip. While the used datasets don't give any information about traffic condition, we added *cluster trip counts* feature which is calculated by summation of the trips for both pick-up and drop-off trips in the cluster related to the trip to represent the number of trips happened in a certain cluster at a definite time. We also separated trips into weekday and weekend. Additionally, the public holidays' traffic varies from normal days so we added *holiday* feature, which takes 1 if the trip day is holiday and 0 if otherwise.

4.1.3 Entity embedding for spatial data

In this paper, we applied entity embedding [9] to the spatial features to make the model understand the dependencies between trips happened in the same cluster. Entity embedding is about mapping categorical variables in a function approximation problem into Euclidean space. Given a set of trip spatial features for each trip as $S = \{\mathrm{PU_{lat}}, \mathrm{PU_{long}}, \mathrm{DO_{lat}}, \mathrm{DO_{long}}, \mathrm{PU_{cls}}, \mathrm{DO_{cls}}, \mathrm{PU_{grid}}, \mathrm{DO_{grid}}\}$ where ($\mathrm{PU_{lat}}$, $\mathrm{PU_{long}}$) denotes pick-up latitude and longitude, ($\mathrm{DO_{lat}}$, $\mathrm{DO_{long}}$) denotes drop-off latitude and longitude, $\mathrm{PU_{cls}}$ denotes pick-up cluster, $\mathrm{DO_{cls}}$ denotes drop-off cluster, PU_{grid} denotes pick-up grid, and $\mathrm{DO_{grid}}$ denotes drop-off grid, this layer then generates the output $\Re$ using approximation function as $\Re = f(S)$.

To learn this function *f(S)*, we first mapped each spatial categorical variable to a vector as follows: c_i: $s_i \rightarrow S_i$. This mapping is identical to an additional layer of linear neurons on the one-hot encoded input. Then, the output of the additional linear layer taking the input c_i is determined as

$$S_i = \sum_{\alpha} \omega_{\alpha\beta}\delta_{S_i\alpha} = \omega_{S_i\beta} \tag{1}$$

where $\omega_{\alpha\beta}$ in (1) is the weight linking the embedding layer with index β to one-hot encoding layer, and $\delta_{S_i\alpha}$ denotes Kronecker delta and the possible values for α are the same as S_i.

4.1.4 Cyclic encoding for temporal features

Cyclic variables usually appear while representing temporal features that are recurrent in their nature as the day of the week or hour of the day. The main attention when dealing with such features is how to encode them in a way where the model can still know that those features occur in cycles. We chose the day of the month, day of the week, and hour of the day to represent the temporal feature. The hour value for a week is between 0 and 23, repeating 7 times. For that, we first give the hour of the day value between 0 and 23, the day of the week value between 0 and 6, and the day of the month value between 0 and 30. To encode this cyclic temporal feature, we applied a sine and cosine transformation as follows:$x_{\sin} = \sin\left(\frac{2 * \pi * x}{\max(x)}\right)$, $x_{\cos} = \cos\left(\frac{2 * \pi * x}{\max(x)}\right)$,where x is an original feature value (i.e., the value of a cyclical feature such as *pickup hour*, *day of the week*, ...etc.). $x_{\sin}$, $x_{\cos}$ are the sine and cosine encoded values of the feature x, respectively. max(x) is the maximum value in the feature range (i.e., the trip feature hour is in range [0, 23] and the maximum value is 23, min is in range [0, 59] and the maximum is 59, *day of the week* is in range [0, 6] and the maximum is 6, ...etc.). The sine encoding is not enough alone as within one cycle two separate time-stamps will get the same sine encoding. To handle this, we applied cosine transformation to represent the sine offset phase and give a unique value within a cycle in two dimensions as shown in Fig. 2, in this figure we can see an example of the sine (y-axis) and cosine (x-axis) transformation for two cyclical features (the hour of the day and the day of the week), i.e., each value in the feature range is represented as a point in 2D space with two unique values, one for the sine and the other for the cosine transformation. Features are scaled into a range between [−1, 1] using Min–Max scaler which also assists neural network.

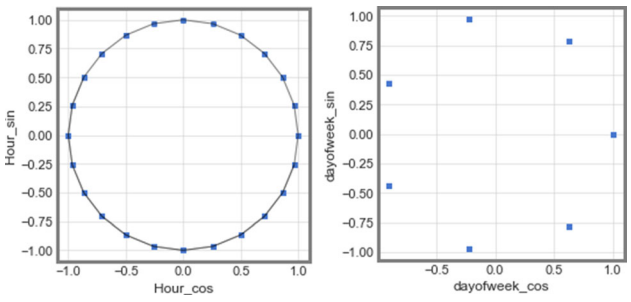


Fig. 2 Sine and cosine transformation for hour of the day and day of the week

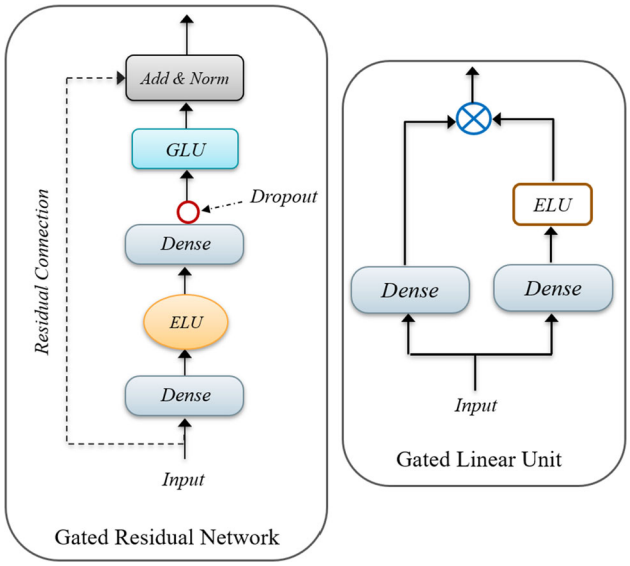


Fig. 3 GRN and GLU structure

4.2 Feature selection module

In travel time prediction, specific relation and contribution of trip features to the target trip duration are typically unknown and complicated to figure out. In this paper, a feature selection module with a gating mechanism illustrated in Fig. 3 to select and pass the most relevant features to the output was developed. This module makes use of a gated residual network [30, 27] with a combination of the gated linear unit. We employed this mechanism in an integration unit (see Sect. 4.3) to process each input trip by computing its feature weights based on their contribution and relation to the output and then select the most relevant features with higher weights. The main structure of the used gating mechanism as shown in Fig. 3 consists of dense layers followed by ELU activation function with drop-out layer and residual connection to add and normalize the shifted input. A gated linear unit [9] with two dense layers and ELU activation function is utilized in this mechanism to control the information, which will be passed to the next layer, by applying element-wise Hadamard product between the input and the weight. By doing this, GLU will help to manage the scope to which the GRN participates in the input. Following we describe tasks for each component of this feature selection module in detail:

GRN controls the information flow and gives the model flexibility to select the most pertinent features by applying nonlinear processing only whenever it is necessary. It takes a vector of input feature χ and generates the output Θ as follows:

$$\Theta(\chi) = \text{LayerNorm}(\text{GLU}(\mu) + \chi) \tag{2}$$

$$\mu = \omega_1 \delta + b_1 \tag{3}$$

$$\delta = \text{ELU}(\omega_2 \chi + b_2) \tag{4}$$

$$f(z) = \begin{cases} z, & \text{if} z > 0. \\ \alpha(\exp(z) - 1), & \text{if } z \leq 0. \end{cases} \tag{5}$$

where *LayerNorm* is used to reduce the training time by computing the mean and variance used for normalization from all of the summed inputs to the neurons in a layer on a single training case [5]. The Exponential Linear Unit (ELU) is an activation function for neural networks. Similar to batch normalization, ELU activation function in Eq. (5), moves mean unit activation closer to zero by its negative values, which accelerates learning by fetching the ordinary gradient closer to the natural gradient because of a reduced bias shift effect [6]. μ and δ are intermediate layers. ω_1, b_1, ω_2, b_2 are weights and biases, z is the input for ELU activation function and α is a hyperparameter to controls the value to which an ELU saturates for negative inputs [6]. We use layer normalization and residual connection by adding input feature vector χ in Eq. (2) to let the model learn the identity mapping by skip connection which makes it easier to optimize the residual mapping than optimizing the original unreferenced mapping. Moreover, these identity skip connections do not append any additional parameters or computational intricacy [16].

GLU in Fig. 3 is used in our task to control the information which goes to the next layer in GRN. It permits the chosen features that are significant for predicting travel time only. The GLU in Eq. (6) takes A as input and yields:

$$\text{GLU}(A) = \text{ELU}(\omega_3 A + b_3) \odot (\omega_4 A + b_4) \tag{6}$$

$$\text{GLU}(A) = \text{ELU}(\gamma_1) \odot (\gamma_2) \tag{7}$$

$$\begin{aligned} \nabla[\text{ELU}(\gamma_1) \odot (\gamma_2)] = \nabla(\gamma_2) \odot \text{ELU}(\gamma_1) \\ + \text{ELU}^{'}(\gamma_1)\nabla(\gamma_1) \odot (\gamma_2) \end{aligned} \tag{8}$$

where $\odot$ is the element-wise Hadamard product and ω_3, ω_4, b_3, b_4 are weights and biases. GLU lets the model manage the scope to which the GRN participates to the original input χ, stepping over the layer completely if necessary as the GLU outputs could be all close to 0 in order to surpress the nonlinear contribution. In (7), γ_1 denotes $(\omega_3 A + b_3)$ and γ_2 represents$(\omega_4 A + b_4)$. While in LSTM the gradient problem is more acute because of the downscaling factors in its *tanh* and *sigmoid* activation function, GLU gradient in (8) contains a channel $\nabla(\gamma_2) \odot ELU(\gamma_1)$ without downscaling, letting the gradient pass through layers while saving the nonlinear qualifications [9].

4.3 Spatial–temporal attention

We design a new pair-wise attention mechanism shown in Fig. 4. In this figure, there are two parts with an integration unit. The integration unit is composed of two GRNs, ELU, product function, and MLP. The first part of the figure is the spatial attention block, while the second part is the temporal attention block. Both parts use the integration unit to fuse the spatial or temporal features in the input and compute their weights based on a self-attention mechanism. Finally, the add and norm layer with a GLU and fully connected network are applied to learn the spatial–temporal attention weights. This spatial–temporal attention block has two attention components interacting with each other to infer the attention weight for the input features. The spatial attention and temporal attention are designed to capture dependencies and correlations in the short and long term between features and their travel time targets as well as similar historical trips (R_T), to find related historical trips in case of data sparsity we look for neighboring trips with a nearby origin and destination using trip grid and trip cluster features that we have extracted (see Sect. 4.1.2).

An integration unit is designed to fuse the related spatial $\Theta(\Gamma_S)$ and temporal $\Theta(\Gamma_T)$ features in the input $\Theta(\Gamma)$ of each component. This integration unit, which makes use of GRN and GLU gating mechanisms to process trip features by computing their weights based on their relation and contribution to the output and then select the ones with higher weights, uses element-wise Hadamard product and multilayer perceptron after applying GRN and so it helps to control the information flow and gives the flexibility to select only the features which are related to each target. In GRN, when $\omega_2 \chi + b_2$ in Eq. (4) is bigger than 0, the ELU function performs as a matching role and when it is smaller than 0, the ELU function produces a fixed output, resulting in linear layer attitude. In this way, nonlinear processing will be applied only where needed and features that will be passed to the next level will be the most relevant features. Features weights are computed in Eqs. (10) and (12) by feeding related trip features through GRN followed by ELU layer for both spatial and temporal parts, respectively. Then, we apply the dot-product in Eq. (11) between the original input and the computed weight vector. For each input, an additional layer of nonlinear processing is employed by feeding each weights vector to GRN. We can see that each feature has its own shared weight. Processed features are then weighted by their feature selection weights and combine to be passed to the next layer.

We use a dot-product function to project each feature as a key to a query, which is all vectors. For each query, it computes a set of attentional weights using a compatibility function between the query and the set of keys. The output

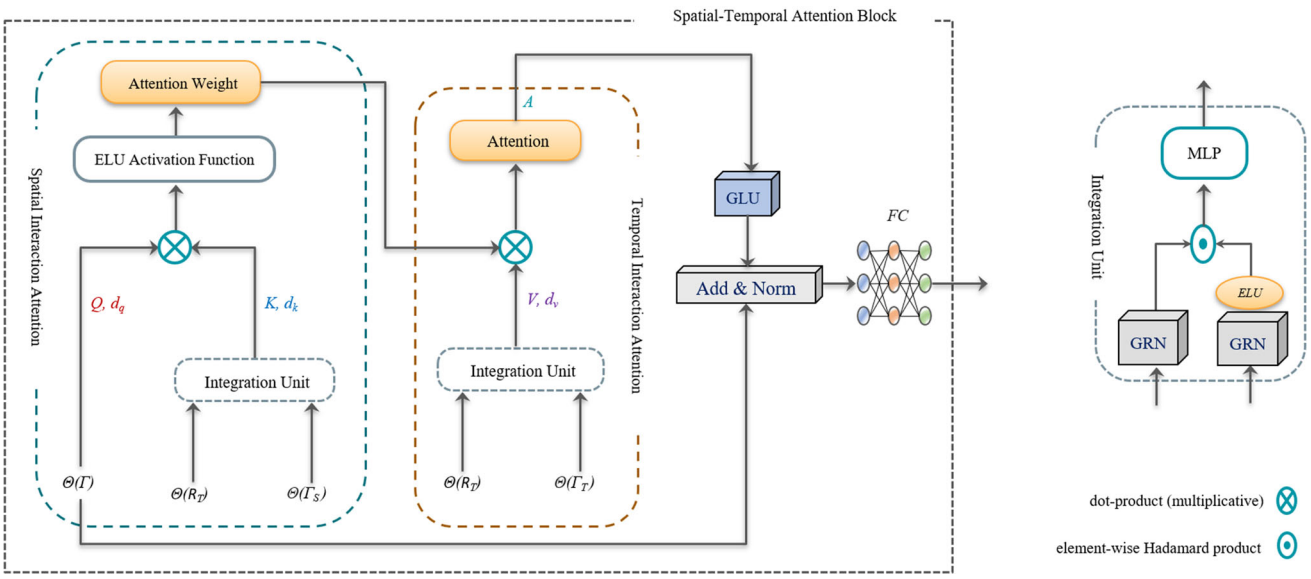


Fig. 4 Spatial–temporal attention block

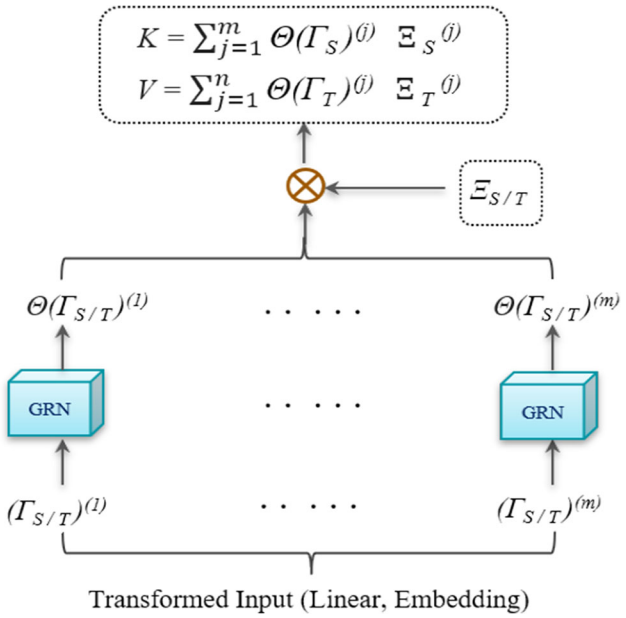


Fig. 5 Feature selection process

feature is then the sum of values weighted by the attention, plus some linear transformations as shown in Fig. 5, where each transformed feature (Γ_S for spatial features or Γ_T for temporal features) with linear and embedding is passed through GRN to get ($\Theta(\Gamma_S)$ for spatial or $\Theta(\Gamma_T)$ for temporal) and then output (K in spatial attention or V in temporal attention) is the sum of each value from feature selection weights vector (spatial vector Ξ_S or temporal vector Ξ_T) with its related value from linear transformed input vector ($\Theta(\Gamma_T)$ for spatial and $\Theta(\Gamma_T)$ for temporal)

$$K = \sum_{j=1}^{m} \Theta(\Gamma_S)^{(j)} \Xi_S^{(j)}, V = \sum_{j=1}^{n} \Theta(\Gamma_T)^{(j)} \Xi_T^{(j)}$$

where m, n is the number of spatial and temporal features, respectively. To accelerates weight learning, we then use the ELU activation function which fetches the ordinary gradient closer to the natural gradient by reducing the bias shift effect. After computing the spatial attention weight, we forward this weight to the temporal attention unit, which uses an integration unit to fuse the temporal features in the same way we discuss previously and then pass the most relevant temporal features only. Finally, this value-pair is projected with the attention weight, which is computed and forwarded from the spatial attention part, using dot-product and generate the final attention value.

In our task, we indicate the input as Γ where each row is a feature vector and then apply linear projection to get the set of queries, keys, and values as follows:

$$Q = \Gamma\omega^Q, K = \Gamma\omega^K, V = \Gamma\omega^V \quad (9)$$

where $\omega^Q, \omega^K, \omega^V$ are weight matrices.

To compute Q in our task, we use Eq. (9). For any trip between two locations, it is obvious that the most important information is location and time information. Spatial information is essential to distinguish between trips that happen at the same time. To capture the spatial correlation in spatial interaction attention unit, we first pass separated spatial feature vector Γ_S to GRN block and compute the output using Eq. (2), then we apply element-wise Hadamard product on this output, where the output comes from applying ELU activation function to other GRN block that takes Γ as whole features vector input in Eq. (11). We then use a two-layer perceptron to flatten the output Ξ_S. The

scaled dot product is then applied to get the attention weight in Eq. (13).

$$\Xi_S = \mathrm{ELU}(\Theta(R_T)) \odot (\Theta(\Gamma_S)) \tag{10}$$

$$\aleph_S = \mathrm{MatMul}(\mathrm{MLP}(\Xi_S), \Gamma) \tag{11}$$

$$\Xi_T = \mathrm{ELU}(\Theta(R_T)) \odot (\Theta(\Gamma_T)) \tag{12}$$

$$A = \mathrm{ELU}\left(\frac{\aleph_S}{\sqrt{d_k}}\right)\Xi_T \tag{13}$$

where R_T is the related historical trips, which are selected based on their location and time period, Ξ_T in Eq. (12) denotes the temporal correlation and compute the same way described above using separated temporal feature vector Γ_T. Finally, we compute the attention value A with the scaled dot and ELU function in Eq. (13).

4.4 Multi-head attention

Implementing the attention mechanism in parallel has offered many advantages over doing a single attention function with keys, values, and queries. In our model, we have adapted the multi-head attention in [32], which was developed for sequence modeling, to our problem domain by first replacing the softmax function with ELU in Eq. (13), ELU function imposes the attention scores of all other features to sum to 1, which is a logical proposition in our case, as this sum is anticipated to be small when the target feature doesn't have correlations with others, and big when it has powerful correlations. Besides, we concatenated and linearly transformed the output matrices of multiple heads into the model dimension using an additional weights matrix as follows:

$$\mathrm{MultiHead}(Q, K, V) = [H_{(i)} | i = 1, 2, ..., m]\omega_H \tag{14}$$

$$H_{(i)} = \mathrm{Attention}(Q\omega^Q_{(i)}, K\omega^K_{(i)}, V\omega^V_{(i)}) \tag{15}$$

$\omega^Q_{(i)}, \omega^K_{(i)}, \omega^V_{(i)}$ are weights for keys, queries and values for specific head, and ω_H combines outputs concatenated from all heads linearly.

By adapting a multi-head attention to our task, we can extend the GSTA capability to focus on different features correlations in different positions as well as it grants the attention layer multiple "representation subspaces" that comes from having multiple sets of Query, Key, and Value weight matrices instead of one in single-head attention. In our model, after many refined trials, we set the number of heads to 4. Every set was initialized at random. After training, every set is utilized to project the input from the previous layer into a different representation subspace.

5 Experimental results and analysis

To test the performance of our model, we used two large real-world datasets, taxi trip dataset from New York and Chengdu. Data cleaning and pre-processing were described in detail in Sect. 4.1. We separated each dataset into train set (60% of trips), validation set (20% of trips), and test set (20% of trips).

5.1 Experiment settings

5.1.1 Compared methods

To compare our model with others work, we select the following models which are related to OD travel time prediction. All models that were used for comparison have been re-implemented along with our proposed method on the same conditions

- **SVR**: The basic idea of Support Vector Regression is to find a formula that can properly predict future values by minimizing error and identifying the hyperplane that optimizes the margin. SVR uses a formula to map the training data from the input space into a higher-dimensional space and then creates a separate hyperplane in the feature space with the maximum margin [11].
- **LGBM**: Light Gradient Boosting Machine is a distributed gradient boosting framework proposed by Microsoft [20]. This model uses a histogram-based method to reduce the impact of high-dimensional data, speed up computation, and avoid overfitting in the prediction model. The continual floating-point eigenvalues are converted to 1 integers and a histogram form with depth limitations and k width is built using this boosting approach.
- **GBDT**: Gradient Boosting Decision Tree finds the best tree mixture by iteratively building base models using a weighted copy of the training data. Each stage in the process of creating a new basis model attempts to fix the errors caused by the earlier base models. We use the GBDT in [40] with the same structure.
- **XGB-IN**: Isolated XGBoost regression model proposed in [19] to characterize the group of journeys that has diverged from majority of journeys which were supposed to pursue the shortest path between pickup and drop-off locations. It mainly based on XGBoost model, which is a decision-tree-based ensemble ML algorithm that uses a gradient boosting framework.
- **MLPTTE**: This method employs a DNN model that contains a 5-layer perceptron with ReLU activation to predict the trip duration. As in [37], MlpTTE's input is nearly similar to GBDT's, with the exception that

categorized values are appropriately embedded into real vectors. The size of the hidden layers is set to 512, 1024, 1024, 512, and 256, respectively. Batch normalization and L2 regularization were also applied.

- **TTE-Ensemble**: Which is proposed in [40] based on an integrated approach. It obtains the features from multi-modality data and uses them as input. Then, it employs GBDT to handle low-dimensional features and the DNN model to treat high-dimensional ones.
- **ST-NN**: Spatio-Temporal Neural Network is a unified deep learning model proposed in [18]. This model employs the "Dist DNN Module" to forecast the trip distance between an origin and a destination GPS point, then applies the "Time DNN Module" to merge this prediction with the time of day as a function to estimate the travel time.
- **FMA-ETA**: FMA-ETA is a deep learning model proposed in [29] that uses a feed-forward network (FFN) to extract spatial–temporal correlations from large data sequentially. To estimate the time of arrival, a novel Multi-factor Attention method was also developed, which efficiently learns the time dependence and contextual relations among time intervals in the sequence.

5.1.2 Evaluation metrics

Five widespread metrics: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Median Absolute Error (MdAE), Root Mean Square Error (RMSE), and R-squared (R^2) are used in the experiments to evaluate and compare the performance of the models.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{16}$$

$$\text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{17}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{18}$$

$$\text{MdAE} = median(|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|) \tag{19}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2} \tag{20}$$

$$\bar{y}_i = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{21}$$

where y_i, $\hat{y}_i$ denote the ground truth and the predicted duration for trip i, respectively, and $\bar{y}_i$ is the mean of the ground truth and calculated in Eq. (21).

5.1.3 Hyper parameters

We adjust hyper-parameter using grid search to detect the optimal parameters. Batch-size = (128, 256, 512), learning rate = (0.1, 0.01, 0.001, 0.0001), dropout = (0.0, 0.1, 0.2, 0.3, 0.4), optimizer = (SGD, Momentum, Adam), and number of head for multi-head attention = (3, 4, 5, 6), we get the optimal parameters that give the highest performance of the model on the validation set. They are as follow: batch-size is 256, learning rate is 0.001, dropout is 0.2, optimizer is Adam, and the number of heads is 4.

5.2 Computational cost

Across both datasets, experiments were performed on a NVIDIA GTX 1080 Ti GPU. Table 3 illustrates the average training time (the average time to train the deep learning model on a single epoch of the whole training data using batch size = 256 or a single iteration for traditional machine learning models), and the average computation time (the average time to predict travel time for 1000 trips in the test data). All results in Table 3 are measured in seconds. From this table, we can see that when the model becomes more complex, the computation time increases. SVR takes the maximum training computation time since the complexity is more than quadratic with the number of dataset records, which makes it hard to scale to big datasets. TTE-Ensemble also takes a long training computation time since it depends on the combination of two models. Although our model took longer training time (892.47 s/epoch, 142 ms/step) than the remaining models due to the complexity of the model structure, yet the complexity in the model structure has helped the model to capture complex and nonlinear spatial–temporal dependencies and

Table 3 Comparison of computational costs of all methods on both datasets

Method	NYC		Chengdu	
	Training (1 epoch)	Predicting (1K trip)	Training (1 epoch)	Predicting (1K trip)
SVR	1154.16	0.139	75.76	0.136
GBDT	583.47	0.134	35.81	0.133
LGBM	550.14	0.131	33.15	0.130
XGB-IN	566.52	0.132	34.97	0.131
MLPTTE	627.68	0.135	39.49	0.134
TTE-Ensemble	921.12	0.151	58.34	0.142
ST-NN	876.35	0.138	54.17	0.136
FMA-ETA	862.91	0.137	52.87	0.135
GSTA	892.47	0.135	55.38	0.133

correlations, and give more accurate predictions than other models. Since the size of the dataset plays an important role in the computation time, we can notice the difference in computation time between two datasets, where the NYC dataset has 130 million trip records while the Chengdu dataset has 8 million trip records.

5.3 Results

Tables 4 and 5 show the GSTA results compared to other models on NYC and Chengdu datasets, respectively. Values with bold font are the best score for each metric. Among the models in two tables, FMA-ETA, ST-NN, and TTE-Ensemble are deep learning models proposed recently, while SVR, LGBM, GBDT, and XGB-IN are ML methods commonly used in travel time prediction. From Tables 4 and 5, we can observe that GSTA outperforms the compared models in all used metrics for both NYC and Chengdu datasets. Results in Tables 4 and 5 show the superiority of our model compared to the FMA-ETA where MAE in our model is reduced by 8.44 s on NYC and 7.62 s on Chengdu. Also, our model MAPE is reduced by 0.72% on NYC and 1.28% on Chengdu. Despite the use of multi-factor attention in FMA-ETA, which concentrates on factors such as speed, link duration, and distance, it neglects the temporal features which are the most important factors to be considered to distinguish between trips that take place in different periods. This importance comes from the fact that the trip's duration in peak hours is much longer than in off-peak hours as a result of traffic congestion. On the contrary, our STA attention mechanism realizes the great effect of temporal features and so employs them side by side with spatial features to capture the complicated dependencies.

Compared to ST-NN, our model MAE is reduced by 12.72 s on NYC and 28.26 s on Chengdu. Also, our model MAPE is reduced by 1.01% on NYC and 2.82% on Chengdu. Even though ST-NN gives rather good results, the DNN models that predict the trip distance first and then employ it to predict the travel time are not well equipped to conclude the intricate correlations from historical trips. While in our model, we use prepossessed features with a more sophisticated and convenient model which employs a mixture of well-compatible components to provide more accurate performance. These components are represented by feature selection and gating mechanism with pair-wise spatial–temporal and multi-head attention to detect complex contextual dependencies.

Table 4 Performance of all models on NYC

Model	MAE(s)	MAPE	RMSE	MdAE	R^2
SVR	292.81	29.47%	409.56	172.68	0.712
GBDT	242.20	26.39%	353.39	120.50	0.815
LGBM	227.66	25.86%	318.76	124.31	0.849
XGB-IN	207.26	23.08%	286.82	111.79	0.867
MLPTTE	169.96	18.53%	221.25	85.54	0.903
TTE-Ensembel	152.71	16.29%	196.68	77.96	0.925
ST-NN	145.93	15.59%	184.65	75.68	0.949
FMA-ETA	141.65	15.23%	181.88	74.42	0.956
GSTA	**133.21**	**14.51%**	**174.15**	**70.96**	**0.972**

Table 5 Performance of all models on Chengdu

Model	MAE(s)	MAPE	RMSE	MdAE	R^2
SVR	350.50	29.73%	455	197.96	0.723
GBDT	285.21	27.03%	403	176.14	0.759
LGBM	275.15	26.63%	390	164.08	0.779
XGB-IN	270.64	26.13%	382	153.21	0.815
MLPTTE	245.37	24.63%	351	131.74	0.856
TTE-Ensembel	221.51	22.73%	328	108.74	0.887
ST-NN	216.17	22.25%	317	101.86	0.918
FMA-ETA	195.53	20.71%	302	82.37	0.926
GSTA	**187.91**	**19.43%**	**290.2**	**76.88**	**0.940**

In comparison with TTE-Ensemble that combines ML and DNN techniques, our model gives better results in terms of MAE where it is reduced by 19.5 s on NYC and 33.6 s on Chengdu, MAPE was reduced by 1.78% on NYC and 3.3% on Chengdu. While the TTE-Ensemble gives good results in the case of linear features interactions and simple dependencies, it doesn't fit well in the case of nonlinear features interactions and sophisticated dependencies where it doesn't show promising results. Our model also outperforms ML approaches since MAE is reduced between 74.05 s $\sim$ 159.6 s on the NYC dataset and 82.73 s $\sim$ 162.59 s on the Chengdu dataset.

It can be seen from Tables 4 and 5 that SVR has the worst results among all models. This is expected because SVR does not distinguish between features in terms of importance but merely attempts to discover the linear relation and approximate mapping from an input domain to the target. In contrast, deep learning models take into account the importance of spatial and temporal features and so they give better results than ML techniques. Yet, some of them are unable to catch complicated nonlinear spatial–temporal feature interactions and correlations. Taking into consideration the previous observations and the importance of spatial–temporal dependencies, the STA attention mechanism plus the multi-head attention and feature

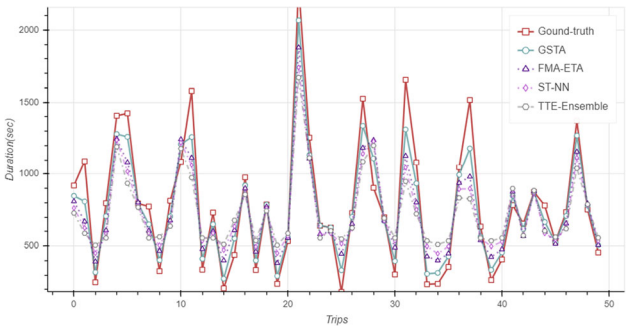


Fig. 6 Prediction vs. ground truth on NYC

selection push our model to perform better than all different models discussed above.

Figure 6 shows a comparison between the ground-truth values and the predictions of all models on the NYC dataset. We selected 50 random trips from the test data. The x-axis represents the number of trips, while the y-axis represents the travel time duration in seconds. The red line represents the real travel time values for those trips, while the green one represents our model's predictions. From this figure, we can see that our model outperforms the other methods and produces predictions close to ground-truth. On the other hand, Fig. 7 shows the ground truth and predicted travel time for all models on the Chengdu dataset using 50 random trips from the test data. We can notice that our model predictions are more accurate than other models.

To further emphasize the competence of our model, we test models in extreme weather conditions on NYC dataset. According to the national weather service in NYC [28], strong wind, snow shower, and freezing rain events happened on the 3rd and 4th of April in 2016. Heavy snow, freezing rain, heavy rain, and damaging wind events happened on days 5, 15, 16, 24, 25 of Feb in 2016.

Table 6 presents the results of models when tested with severe weather on NYC dataset. Despite the abnormal data, our model gives the best results and outperforms other models. Figure 8 illustrates the real travel time compared

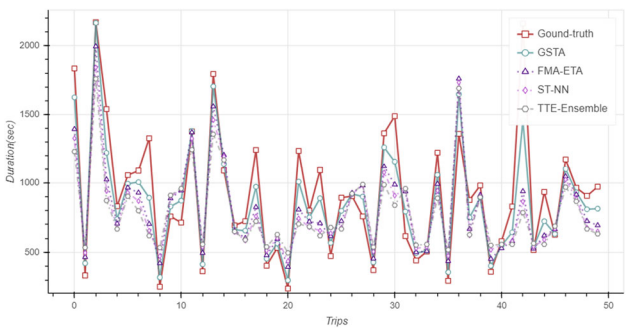


Fig. 7 Prediction vs. ground truth on Chengdu

Table 6 Performance of models with severe weather on NYC

Model	MAE(s)	MAPE	RMSE	MdAE	R^2
LGBM	319.61	27.67%	394.98	158.93	0.601
XGB-IN	301.17	26.34%	388.41	149.13	0.635
MLPTTE	285.11	23.71%	351.52	127.26	0.766
TTE-Ensembel	224.84	21.23%	334.17	118.78	0.831
ST-NN	219.63	20.04%	325.54	112.07	0.853
FMA-ETA	214.87	19.53%	319.73	98.52	0.861
GSTA	**201.34**	**18.92%**	**305.88**	**94.24**	**0.879**

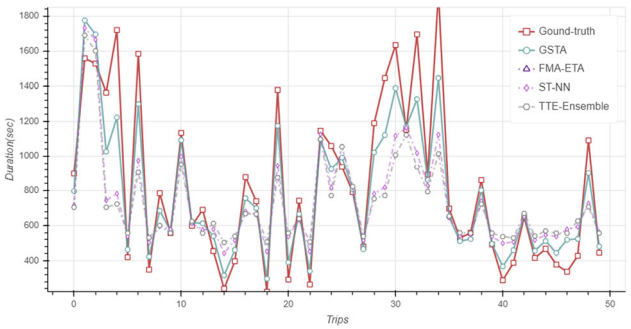
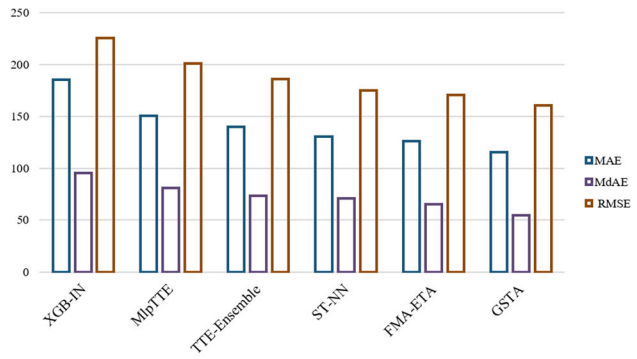


Fig. 8 Prediction vs. ground truth in severe weather conditions on NYC

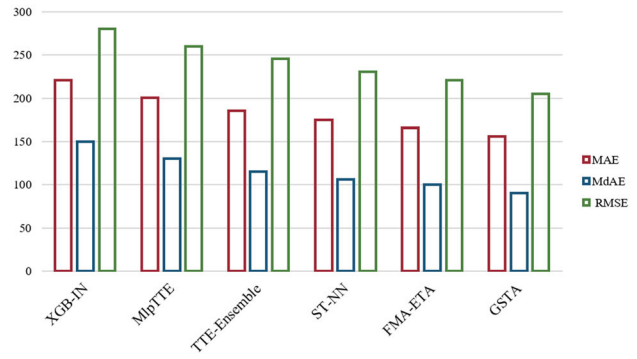
to the predictions of all models in extreme weather conditions on the NYC dataset. The results of 50 randomly selected trips from the trips that happened during abnormal weather demonstrate the superiority of our model performance under severe weather conditions compared to other methods.

Moreover, we test our model on the two datasets during an off-peak period (12:00 ∼ 15:00) and two different peak periods (7:00 ∼ 10:00) and (17:00 ∼ 20:00). Since the traffic volume in peak periods is much higher than in off-peak, the MAE and RMSE in peak periods are larger than in off-peak periods.

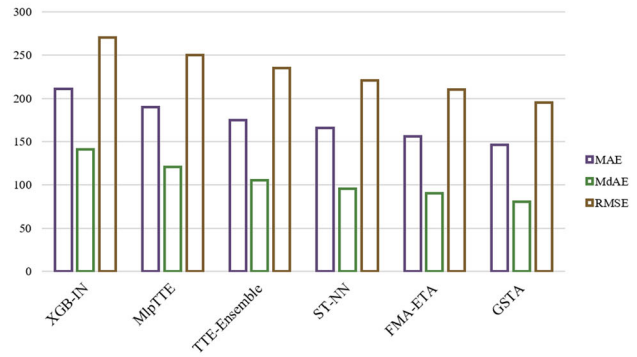
Figure 9 illustrates a comparison between all models' prediction errors at different periods of the day in terms of MAE, MdAE, and RMSE in the NYC dataset. Figure 9a shows all models' prediction errors during the off-peak period. Figure 9b, c show the comparison between different models' prediction errors during the morning-peak and afternoon-peak periods, respectively. On the other hand, Fig. 10 presents a comparison of all models' prediction errors at different periods of the day in terms of MAE, MdAE, and RMSE in the Chengdu dataset. Figure 10a displays all models' prediction errors during the off-peak period. Figure 10b, c show the comparison between different models' prediction errors during the morning-peak and afternoon-peak periods, respectively. The results



(a) Off-peak period 12:00∼15:00

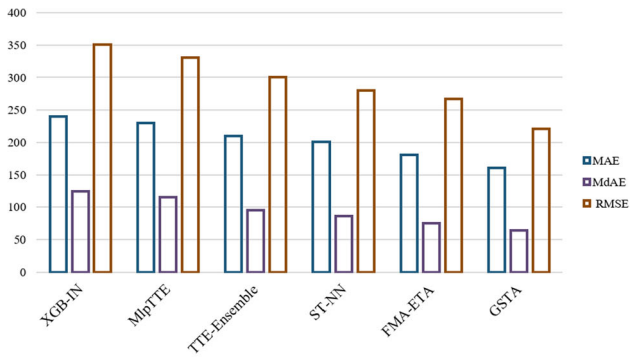


(b) Peak period 7:00∼10:00

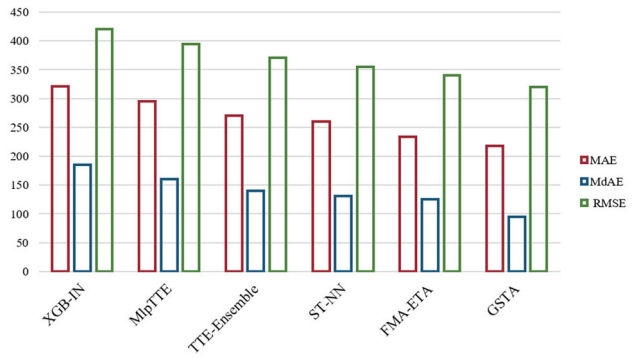


(c) Peak period 17:00∼20:00

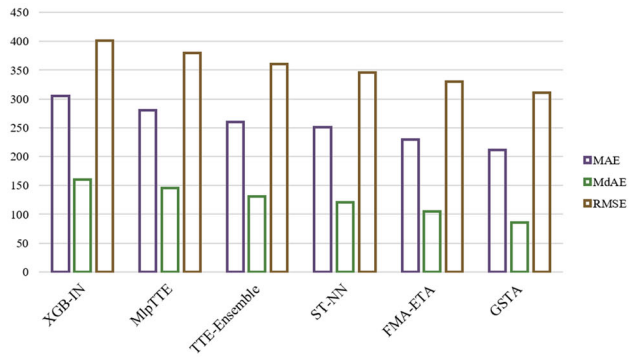
Fig. 9 Peak vs. off-peak performance of all models on NYC



(a) Off-peak period 12:00∼15:00



(b) Peak period 7:00∼10:00



(c) Peak period 17:00∼20:00

Fig. 10 Peak vs. off-peak performance of all models on Chengdu

demonstrate the better performance of our model compared to all other models at different periods of the day.

5.4 Ablation analysis

Spatial–temporal attention, feature selection module, and multi-head attention are the main components of GSTA. To check the contribution of each component, we conduct some ablation studies by removing each part as follows:

1. **Without-STA**: we remove the spatial–temporal attention component. So, we pass the concatenated input Γ to the fully connected network after applying GLU and layer normalization.
2. **Without-FS**: we remove the feature selection component by replacing the integration unit and gating mechanism components with a simple two-layer MLP that applies the ELU activation function.
3. **Without-MHA**: we remove the multi-head attention component. So, we pass the output of spatial–temporal attention A to GRN to get $\Theta(\chi)$ then through the remaining parts to get the output.

Tables 7 and 8 illustrate the results of the above models on NYC and Chengdu datasets. From these tables, we notice that MAE increases from 133.21 to 215.12 on NYC and

Table 7 Ablation analysis on NYC

Model	MAE(s)	MAPE	RMSE	MdAE	R^2
Without-STA	215.12	23.54%	324.6	118.79	0.852
Without-FS	213.24	23.37%	322.11	115.12	0.868
Without-MHA	167.93	19.61%	280.22	95.53	0.925
GSTA	**133.21**	**14.51%**	**174.15**	**70.96**	**0.972**

Table 8 Ablation analysis on Chengdu

Model	MAE(s)	MAPE	RMSE	MdAE	R^2
Without-STA	265.48	26.63%	377.65	135.76	0.817
Without-FS	260.67	26.41%	368.14	131.98	0.822
Withou-MHA	212.33	22.05%	319.89	114.74	0.912
GSTA	**187.91**	**19.43%**	**290.2**	**76.88**	**0.940**

from 187.91 to 265.48 on Chengdu, in case of removing the spatial–temporal attention (STA). Therefore, we can notice that STA indeed helps to improve the performance of the model. STA has a great effect on the model accuracy and this is because STA can focus on specific trips and features instead of handling all of them equally, and so this contextual pair-wise attention mechanism captures the dynamic correlations as well. We can also see that feature selection with a gating mechanism have also a big effect due to its rule in selecting the most relevant features based on their contributions. We can notice that without feature selection, the result increases from 133.21 to 213.24 on NYC and from 187.91 to 260.67 on Chengdu in terms of MAE. Whereas the multi-head attention helps to infer complex and long-term dependencies, It has a smaller impact compared to other components since it is applied when there is complicated dependencies. In the case of removing MHA, MAE increases from 133.21 to 167.93 on NYC and from 187.91 to 212.33 on Chengdu. Finally, these results confirm that combining all components in one single model gives the best result on both NYC and Chengdu datasets.

6 Conclusion

Travel time prediction plays an important role in managing and regulating traffic flow. In this paper, we proposed a deep learning model by integrating a new pair-wise attention mechanism with spatial inference and temporal reasoning to capture spatial–temporal dependencies. Importantly, a feature selection module with an integration unit and gating mechanism was developed to give the model the flexibility to select the most relevant features based on their contributions to the output. Moreover, we adapted and integrated multi-head attention to improve the model performance in case of sophisticated dependencies in long term. Comprehensive experiments on two large datasets in NYC and Chengdu have been performed, experimental results have shown the efficiency and the superiority of our model in comparison with other models.

For future work, we consider the following trends for developing our model. For example, CNN can be integrated with multimodal deep learning techniques to achieve better results in selecting relevant features and capturing spatial dependencies. Also, we would like to investigate the influence of exterior factors to further enhance prediction accuracy, such as traffic flow, special occasions, and driver behavior.

Acknowledgements This work is supported in part by the National Natural Science Foundation of China under Grant U1811463, and also in part by the Innovation Foundation of Science and Technology of Dalian under Grant 2018J11CY010.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Abbar S, Stanojevic R, Mokbel MSTAD (2020) Spatio-temporal adjustment of traffic-oblivious travel-time estimation. In: Proceedings—IEEE international conference on mobile data management 2020–June, 79–88 https://doi.org/10.1109/MDM48529.2020.00029
2. Abdollahi M, Khaleghi T, Yang K (2020) An integrated feature learning approach using deep learning for travel time prediction. Exp Syst Appl. https://doi.org/10.1016/j.eswa.2019.112864
3. Aceto G, Ciuonzo D, Montieri A, Pescapè A (2019) Mimetic: mobile encrypted traffic classification using multimodal deep learning. Comput Netw 165:106944. https://doi.org/10.1016/j.comnet.2019.106944. https://www.sciencedirect.com/science/article/pii/S1389128619304669
4. Ahmed MS, Cook AR (1979) Analysis of freeway traffic time-series data by using box-Jenkins techniques. Transp Res Rec 722:1–9
5. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. http://arxiv.org/abs/1607.06450
6. Clevert DA, Unterthiner T, Hochreiter S (2016) Fast and accurate deep network learning by exponential linear units (ELUs). In: Proceedings of the 4th international conference on learning representations, ICLR 2016: conference track proceedings. pp 1–14
7. Commission NTL (2021) TLC trip record data: TLC. Accessed on 2 May 2021. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
8. Dataset DCT (2021) DiDi Chengdu taxi dataset. Accessed on 2 May 2021. https://outreach.didichuxing.com/app-vue/dataList

9. Dauphin YN, Fan A, Auli M, Grangier D (2016) Language modeling with gated convolutional networks. CoRR **abs/1612.0**. http://arxiv.org/abs/1612.08083
10. Delling D (2018) Route planning in transportation networks: from research to practice. In: Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems, SIGSPATIAL '18, p. 2. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3274895.3282802
11. Drucker H, Burges CJC, Kaufman L, Smola A, Vapnik V (1996). Support vector regression machines. In: Proceedings of the 9th international conference on neural information processing systems, NIPS'96, p. 155–161. MIT Press, Cambridge, MA, USA
12. Du W, Sun B, Kuai J, Xie J, Yu J, Sun T (2021) Highway travel time prediction of segments based on ANPR data considering traffic diversion. J Adv Transp 2021:1–16. https://doi.org/10.1155/2021/9512501
13. Fang X, Huang J, Wang F, Zeng L, Liang H, Wang H (2020) ConSTGAT: contextual spatial-temporal graph attention network for travel time estimation at Baidu Maps. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining. pp 2697–2705. https://doi.org/10.1145/3394486.3403320
14. Fei X, Lu CC, Liu K (2011) A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction. Transp Res C Emerg Technol 19(6):1306–1318. https://doi.org/10.1016/j.trc.2010.10.005
15. Guo G, Zhang T (2020) A residual spatio-temporal architecture for travel demand forecasting. Transp Res C Emerg Technol 115:102639. https://doi.org/10.1016/j.trc.2020.102639
16. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition 2016-Decem, 770–778. https://doi.org/10.1109/CVPR.2016.90
17. Ishak S, Kotha P, Alecsandru C (2003) Optimization of dynamic neural network performance for short-term traffic prediction. Transp Res Rec 1836:45–56
18. Jindal I, Tony Q, Chen X, Nokleby M, Ye J (2017) A unified neural network approach for estimating travel time and distance for a taxi trip. http://arxiv.org/abs/1710.04350
19. Kankanamge KD, Witharanage YR, Withanage CS, Hansini M, Lakmal D, Thayasivam U (2019) Taxi trip travel time prediction with isolated xgboost regression. In: MERCon 2019: Proceedings, 5th international multidisciplinary moratuwa engineering research conference (April 2020), 54–59 . https://doi.org/10.1109/MERCon.2019.8818915
20. Ke G, Meng Q, Finely T, Wang T, Chen W, Ma W, Ye Q, Liu, T.Y (2017). Lightgbm: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems 30 (NIP 2017)
21. Khodaverdian Z, Sadr H, Edalatpanah SA (2021) A shallow deep neural network for selection of migration candidate virtual machines to reduce energy consumption. In: Proceedings of the 2021 7th international conference on web research (ICWR), pp 191–196. https://doi.org/10.1109/ICWR51868.2021.9443133
22. Khodaverdian Z, Sadr H, Edalatpanah SA, Solimandarabi MN (2021) Combination of convolutional neural network and gated recurrent unit for energy aware resource allocation. CoRR abs/210612178. https://arxiv.org/abs/2106.12178
23. Li Y, Fu K, Wang Z, Shahabi C, Ye J, Liu Y (2018) Multi-task representation learning for travel time estimation. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining
24. Myung J, Kim D, Kho S, Park C (2011) Travel time prediction using k nearest neighbor method with combined data from vehicle detector system and automatic toll collection system. Transp Res Rec 2256:51–59
25. Ngiam J, Khosla A, Kim M, Nam J, Lee H, Ng AY (2011) Multimodal deep learning. In: Proceedings of the 28th international conference on international conference on machine learning, ICML'11, p. 689–696. Omni press, Madison, WI, USA
26. Prokhorchuk A, Dauwels J, Jaillet P (2020) Estimating travel time distributions by bayesian network inference. IEEE Trans Intell Transp Syst 21(5):1867–1876. https://doi.org/10.1109/TITS.2019.2899906
27. Savarese P, Figueiredo D (2017) Residual gates: a simple mechanism for improved network optimization
28. Service N.W (2021) NWS New York significant weather events archive. Accessed on 2 May 2021. https://www.weather.gov/okx/stormevents
29. Sun Y, Wang Y, Fu K, Wang Z, Yan Z, Zhang C, Ye J (2020) FMA-ETA: estimating travel time entirely based on FFN with attention pp 1–10 . http://arxiv.org/abs/2006.04077
30. Tan K, Chen J, Wang D (2018) Gated residual networks with dilated convolutions for supervised speech separation department of computer science and engineering, the Ohio state university, USA center for cognitive and brain sciences, the Ohio state university, USA. Icassp 1:21–25
31. Ting P, Wada T, Chiu Y, Sun M, Sakai K, Ku W, Jeng AA, Hwu J (2020) Freeway travel time prediction using deep hybrid model: taking sun Yat-Sen freeway as an example. IEEE Trans Veh Technol 69(8):8257–8266
32. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inform Process Syst 2017:5999–6009
33. Wang D, Zhang J, Cao W, Li J, Zheng Y When will you arrive? Estimating travel time based on deep neural networks
34. Wang H, Tang X, Kuo YH, Kifer D, Li Z (2019) A simple baseline for travel time estimation using large-scale trip data. ACM Trans Intel Syst Technol 10(2):1–22. https://doi.org/10.1145/3293317
35. Wang Z, Fu K, Ye J, Labs DAI, Chuxing D (2018) Learning to estimate the travel. Time 1:858–866
36. Wu Z, Rilett LR, Ren W (2021) New methodologies for predicting corridor travel time mean and reliability. Int J Urban Sci. https://doi.org/10.1080/12265934.2021.1899844
37. Xu S, Zhang R, Cheng W, Xu J (2020) MTLM: a multi-task learning model for travel time estimation. GeoInformatica. https://doi.org/10.1007/s10707-020-00422-x
38. Yuan H, Li G, Bao Z, Feng L (2020) Effective travel time estimation: when historical trajectories over road networks matter. In: Proceedings of the ACM SIGMOD international conference on management of data. pp 2135–2149. https://doi.org/10.1145/3318464.3389771
39. Zhang H, Wu H, Sun W, Zheng B (2018) DEEPTRAVEL: a neural network based travel time estimation model with auxiliary supervision. IJCAI Int Joint Conf Artif Intel 2018:3655–3661. https://doi.org/10.24963/ijcai.2018/508
40. Zou Z, Yang H, Zhu AX (2020) Estimation of travel time based on ensemble method with multi-modality perspective urban big data. IEEE Access 8(2):24819–24828. https://doi.org/10.1109/ACCESS.2020.2971008