MCT-TTE: Travel Time Estimation Based on Transformer and Convolution Neural Networks

Fengkai Liu, Jianhua Yang, Mu Li, and Kuo Wang

1College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
2North China Institute of Computing Technology, Beijing 100083, China
3School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Correspondence should be addressed to Mu Li; limu@buaa.edu.cn

Received 6 December 2021; Accepted 11 February 2022; Published 12 April 2022

Academic Editor: Zahid Mehmood

Copyright © 2022 Fengkai Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this paper, we propose a new travel time estimation framework based on transformer and convolution neural networks (CNN) to improve the accuracy of travel time estimation. We design a traffic information fusion component, which fuses the GPS trajectory, real road network, and external attributes, to fully consider the influence of road network topological characteristics as well as the traffic temporal characteristics on travel time estimation. Moreover, we provide a multiview CNN transformer component to capture the spatial information of each trajectory point at multiple regional scales. Extensive experiments on Chengdu and Beijing datasets show that the mean absolute percent error (MAPE) of our MCT-TTE is 11.25% and 11.78%, which is competitive with the state-of-the-arts baselines.

1. Introduction

With the continuous improvement of China’s economic level and the gradual acceleration of the urbanization process, the traffic demand of urban residents also shows a trend of rapid growth. The rapid growth of population in large and medium-sized cities and the continuous increase of the number of motor vehicles pose a great challenge to the carrying capacity of urban traffic. The number of vehicles increased sharply, the density of the road network increased continuously, and the transportation system of the whole city became increasingly complex. How to plan people’s travel and maximize the efficiency of commuting; how to ensure the smooth road network and minimize the resource consumption has become an urgent problem to be solved. Among them, the estimation of travel time for a given path, namely, the travel time estimation (TTE) problem, is a basic problem in path planning, navigation, and traffic scheduling, as discussed in [1].

Although the TTE problem has been extensively studied in the past, providing an accurate travel time is still very challenging, mainly due to the high uncertainty of traffic conditions caused by weather, temporary controls, and unexpected accidents. It is a high challenging work to give a more accurate forecast of travel time considering weather, temporary control and unexpected accidents in real time.

Meanwhile, with extensive deployment of smart sensors across the city network and the increasing presence of high-speed 5G infrastructure, it has become more accessible than ever before to collect cross-domain datasets for optimization and smart planning in a complex traffic scenario, which makes it possible for the deployment of big data analytical model for more accurate travel time prediction under a dynamic cross-domain traffic environment.

In recent years, deep learning has made breakthrough progress in computer vision, natural language processing, and other fields. For traffic data, there are not only time series data similar to the sentence sequence features in natural language processing, but also road network features suitable for graph neural networks. Therefore, the accurate prediction of traffic spatial and temporal data based on deep learning requires aggregation of various fields of technology, which is a challenging task [2]. Recent studies show that the prediction of spatio-temporal traffic data is also transformed
from traditional statistical methods to machine learning and deep learning. Especially in the field of deep learning, due to the strong learning ability of deep learning for massive data, and different network structures have their own applicable scenarios, various neural networks that integrate different components have become the most mainstream and the most accurate method for spatio-temporal data prediction. For example, the design of attention mechanism can effectively capture the temporal correlation before and after traffic data, and the modeling of map convolutional network on the road network can learn the changes of road flow characteristics. However, there are still few relevant researches compared with other directions, and there are many important problems waiting to be solved [3]. Therefore, the traffic spatio-temporal data prediction based on deep learning is not only technically feasible and challenging, but also has a wide range of application values in real life.

In this paper, we propose an end-to-end framework to learn the spatio-temporal patterns and estimate the travel time based on the given path and the corresponding external factors, called MCT-TTE. In the following chapters, we first describe the basic definitions and assumptions used in this paper, then describe the model structure in detail, and finally present the experimental design and compare it with the current popular models.

2. Preliminary

The definition of travel time estimation has been clearly stated in the aforementioned research [4]. We present several preliminaries and define our problem formally here.

2.1. Definition 1: Trajectory. The results and discussion may be presented separately, or in one combined section, and may optionally be divided into headed subsections.

The trajectory \( T = \{p_1, \ldots, p_{|T|}\} \) is defined as a sequence of consecutive historical GPS points. Each GPS point \( p_i \) may have a set of properties, including latitude \( p_i.lat \), longitude \( p_i.lon \), timestamp \( p_i.ts \), and status \( p_i.state \). Since the GPS records are usually generated every fixed length time gap, which may cause the learning of the trival pattern, we resampled each historical trajectory with roughly the same distance. Furthermore, each trajectory contains its external factors such as the starting time, the day of the week, the distance. However, there are still few relevant researches compared with other directions, and there are many important problems waiting to be solved [3]. Therefore, the traffic spatio-temporal data prediction based on deep learning is not only technically feasible and challenging, but also has a wide range of application values in real life.

Since the raw trajectory data is collected at a certain time interval with measuring errors, it is difficult to restore the real driving conditions only by the raw trajectory data. In the traffic data preprocessing stage, we combined the raw GPS trajectory with the real road network data using map matching and adopted a road segment embedding method to finally generate the embedded road segment vector sequence.

2.2. Definition 2: Embedded Road Segment Vector. A road network is represented by a directed graph \( G = (V, E) \) where \( V = \{r_1, r_2, \ldots, r_N\} \) is the set of all nodes which mean the road segments (RS for short), and \( E = \{(r_i, r_j) | r_i, r_j \in V\} \) is the set of all edges which mean the connection relation between two RSs. A RS sequence \( RSS = \{r_1, r_2, \ldots, r_n\} \) is defined as an ordered sequence of RSs, generated by map matching technology from a GPS trajectory mapped. The continuous trajectory points on the same RS will be mapped to the same \( r_i \). Due to the lack of relevant information of data source (OpenStreetMap, OSM), we do not consider the change of the road network over time and adopt the static road network structure here. The simple embedding method of RS, such as one-hot coding, will lose the important information such as upstream and downstream relationships. Therefore, we propose a learnable embedding model to generate the Embedded Road Segment Vector (ERSV for short). Each \( r_i \) in the RSS will be embedded to an embedded vector \( r_i.ev_f \) using a certain mapping method \( f \).

2.3. Definition 3: Travel Time Estimation. The problem of Travel Time Estimation (TTE for short) can be transformed into a supervised learning problem by integrating the raw GPS trajectory, the ERSV sequence, and other external attribute features as model inputs, and taking the travel time of the trajectory as the ground true label. Assumed that we have a travel path (represented by the historical GPS trajectory \( T \)) which is specified by the user or generated by the route planning app, our goal is to estimate the travel time from the starting to the destination through the travel path with the corresponding external factors.

3. Model Architecture

The architecture of MCT-TTE is shown in Figure 1. It is composed of three components: traffic information fusion component, multiview CNN Transformer component, and multitask learning component.

The traffic information fusion component includes two parts: information embedding and path encoding. It extracts local spatial feature information through feature pre-extraction of trajectory and fuses the processed results with given external information. The information embedding part embeds the raw trajectory with real road network information and external information. The path encoder captures the spatial structure features of the trajectory, and processes the external factors (e.g., driver ID, weather) and basic information of the given path (e.g., start time, total distance) to generate the fusion feature map, which will be fed to the multiview CNN Transformer component as inputs.

The multiview CNN Transformer is the main component that learns the spatial correlations and temporal dependencies from the fusion feature map. The proposed multiview CNN Transformer component is comprised of a series of multiview CNN Transformer (MCT) blocks. Each MCT block contains a spatial transformer and a temporal transformer to jointly learn spatial-temporal features in the context of dynamically changing dependencies. We improve the network structure of the spatial transformer by adding multiview CNN module to extract multilevel spatial

\[
V \rightarrow \text{traffic information fusion component} \rightarrow \text{multitask learning component}
\]
correlation information, so as to be suitable for traffic travel time estimation problem.

Finally, the multitask learning component estimates the travel time of the given path based on the previous two components, capable of balancing the trade-off between individual estimation and collective estimation.

### 3.1. Traffic information fusion component

Travel times are influenced by many factors. The time of departure, the day of the week, weather factors, traffic control factors, accident situations, and even driving habits all combine to influence the travel time of a particular route. Generally, we can roughly judge the impact of some of these factors on travel time. For example, weekdays are more congested than weekends, rainy days are more congested than sunny days, and roads with traffic restrictions are more congested. However, these factors also have a deeper effect on travel time. These deeper roles require machine learning to dig deeper. Although there are many factors, it is difficult to fully study the interaction of these factors. It is hard to construct a large-scale traffic data set containing various factors, especially weather conditions, traffic emergencies, traffic control information, etc. Although we know that factors such as weekday or weekend and departure dates have a significant impact on travel time, the model will have difficulty learning the deep impact of these factors due to the lack of open big dataset.

In this article, we incorporate the attributes of the driver ID, the time information (day of the week and time slot of travel start), and the weather condition (rainy/sunny/windy, etc.). In particular, we specify Driver_id to indicate the driver ID, Week_id to indicate the day of the week, Time_id to indicate the start time of the trajectory, Weather_D to indicate the weather during the day, and Weather_N to indicate the weather at night. After that, the embedding method are used to transform each categorical attribute into a low-dimensional real vector. Furthermore, we incorporate another important attribute, the travel distance along the path, for the reason that previous relevant research shows that each segmented path distance has an important impact on the estimation of travel time.

As shown in Figure 2, TTE based solely on raw sampled trajectory data (such as evenly sampled trajectory points with equal distance) may lose the turning information of some intersections (for example, P7 — P8 — P9 will be regarded as a road), thus introducing wrong features in the modeling process.

To avoid the loss of accuracy, we map the GPS coordinates to the RSs and then embed to the ERSVs, so as to preserve the upstream and downstream dependence between the RSs. Inspired by the word embedding technology and the processing ideas of neural network language model, we propose a RSV Embedding model, which uses the skipgram model to learn the sequence of RSs, ensuring that the distance between the vectors of the RSs with upstream and downstream relationships is relatively close, while that without upstream and downstream relationships are relatively far. The training of RSV embedding model can be carried out independently of the whole TTE model, that is, the trajectory sequence is input into RSV embedding model in advance for training, so that each RS in the sequence is mapped to a low-dimensional vector ERSV. Before this, make sure the transformations from the raw trajectory sequence to the RSS has been done, which may need to go through the process of map matching using mature tool kits.

Finally, the ERSV will be fused with the raw trajectory and other attributes through the path encoding component to obtain the fusion feature map of traffic information, which will be input into multiview CNN Transformer model for spatial and temporal feature extraction.

#### 3.1.1. RSV Embedding Model

Assumed that we have converted GPS trajectory points into RSS using map matching. Each RS has the corresponding segment number $r_{i, id}$, segment length $r_{i, dis}$ and road speed limit $r_{i, spd}$. However, it is difficult to identify the upstream and downstream dependence between road segments only by the ID of RS. Now we need a RSV embedding method to embed the RS into a low-dimensional vector. The simplest embedding method is One-hot coding, but it fails to describe the correlation of adjacent RSs, and the dimension of the vector will be large as the number of RSs grows. Another simple embedding method may use the center point coordinate of each RS as

![Figure 1: MCT-TTE Architecture.](image-url)
the representation. However, this embedding method can only obtain the adjacent information of RSs and cannot obtain the upstream and downstream relationships.

In our proposed RSV Embedding model, we use the skip-gram method to embed the RS. Skip-gram method comes from the field of word embedding technology, which can embed words into low-dimensional vectors and make the corresponding vectors with similar semantics close to each other [5, 6].

The main idea of the RSV embedding method based on skip-gram is to predict T segments upstream and downstream of a certain RS \( r_i \), which is a multi-label problem essentially. First, we use a sliding window to generate training samples for the RSS, as shown in Figure 3. When the sliding window size equals 2, two pairs of upstream samples and downstream samples will be generated. Here, we extract the upstream and downstream features of RS, respectively.

Then, the pseudo-neighbor segment of the central unit is constructed as the negative sample. The generation of negative samples is based on the weighted sampling of the occurrence frequency of each segment. The higher the occurrence frequency of the RS, the easier it is to be sampled and generate negative samples. The probability of the RS being sampled is as Equation 1. The \( f(r) \) function is used to count the frequency of RSs. Laplace correction is used to prevent some segments from being unsampled due to lack of data [2].

\[
P(r) = \frac{\text{freq}(r) + 1}{\sum_{j=1}^{N} \text{freq}(r_j) + N}
\]

The sample scoring model should be able to give correct judgment to positive and negative samples according to the hidden layer information mapped to, that is, to judge the correct section with upstream and downstream relationships. Our scoring model is simply divided into input layer, hidden layer, and output layer. The input layer is the one-hot code of the central RS and the upstream/downstream RS, which are \( r^u \) and \( r^d \), respectively. The hidden layer \( h \) is calculated by the input layer and the matrix \( W \) using \( h = f(r) = \sigma(W \cdot x + b) \), \( b \) is the offset vector and \( \sigma \) is the sigmoid function. The central, upstream and downstream RS are mapped to the hidden layer \( h^c, h^u \) and \( h^d \) with different parameters \( W^c, b^c, W^u, b^u, \) and \( W^d, b^d \) respectively. The score \( y_{\text{score}} \) is calculated as (\( (.)^T \) represents the transpose operation):

\[
y_{\text{score}} = \sigma((h^c)^T h^u/d).
\]

The RSV Embedding Model scores the positive and negative samples of the upstream and downstream RSs at the same time with a preset ratio. In each iteration, a positive sample pair and \( k \) negative sample pairs are scored simultaneously using the scoring model. The model needs to maximize the positive sample score while minimizing the negative sample score. Thus, the objective function is:

\[
\max L = \prod_{r \in \text{up/down}(r)} \left( \prod_{r \in \text{Neg}(r)} y_{\text{score}} \prod_{r \in \text{up/down}(r)} (1 - y_{\text{score}}) \right),
\]

where up/down \( (r_j) \) represents the upstream or downstream RSs of \( r_i \), and Neg \( (r_i) \) represents all RSs with negative sampling of \( r_i \).

After training, the RS can be mapped to the ERSV \( r_i, ev^c, r_i, ev^u \) and \( r_i, ev^d \). Therefore, the upstream and downstream dependent information between RSs is now well preserved by the RSV embedding model.

3.1.2. Path Encoder. To convert the raw trajectory data into a fusion feature map that is convenient for transformer model to capture spatio-temporal features and integrate it with additional traffic attributes, we design a simple yet effective component to incorporate such factors into our model, called path encoder. Path encoder component locally captures the spatial structure features of the trajectory and generates the input of the next stage component by combining external features, raw trajectory, road segment sequence, and its statistics (such as departure time and road length, etc.).

By map matching and RSV embedding, we attach each GPS location point \( p_i \) with the center, upstream and downstream ERSV \( p_i, ev^c, p_i, ev^u \) and \( p_i, ev^d \) as well as the length of RS \( p_i, \text{dis} \) and speed limit \( p_i, \text{spd} \) of the corresponding RS. Note that there may be multiple GPS points corresponding to the same RS by using map matching. Here, we retain the length of the raw GPS trajectory and supplement the ERSV as redundant information to avoid the loss of potential deep information. Then, for each GPS point

**Figure 2: Road network.**

**Figure 3: Example of sliding window model when the size of sliding window equals 2.
\( p_i \) in the sequence, we use a nonlinear mapping to map the
i-th GPS point into vector \( \text{traj}_i \in \mathbb{R}^{128} \), as shown in (4),
where the superscript em indicates the embedding vector, \( \circ \)
indicates the concatenate operation, \( W_{\text{loc}} \) is a learnable
weight matrix and \( \mathbb{R} \) represents the set of reals.

\[
\text{traj}_i = \tanh \left( W_{\text{loc}} \cdot \left[ p_i, \text{lon} \circ p_i, \text{lat} \circ p_i, \text{state}^\text{em} \circ p_i, \text{ev}^\text{em} \circ p_i \circ p_i, \text{dis} \circ p_i, \text{spd} \right] \right). \tag{4}
\]

Thus, the output sequence \( \text{traj} \in \mathbb{R}^{128 \times |T|} \) represents the
nonlinearly mapped locations, and each channel of describes the
geographical feature of the trajectory.

In the same way, we incorporate the external attributes
to generate the external vector \( \text{attr} \in \mathbb{R}^{128} \) as follow:

\[
\text{attr} \in \tanh \left( W_{\text{attr}} \cdot \left[ \text{Dis} \circ \text{Week}^\text{em}, \text{Time}^\text{em}, \text{Driver}^\text{em} \circ \text{Weather}^\text{D}^\text{em} \circ \text{Weather}^\text{N}^\text{em} \right] \right),
\tag{5}
\]

Dis represents the total distance of the trajectory. We divide
one day into 1440 timeslots and each timeslot corresponds to
one minute, which is represented by Time_id. All factors
that are categorical values which can not feed to the neural
network directly are embedded using the simple embedding
method. Since the same external information is suitable for
all GPS points of a trajectory, the vector attr does not need
the subscript \( i \) here. In fact, if other attributes are available,
they can also be concatenated after embedded. The fusion
method proposed here is not limited to these specific at-
tributes, but can be extended to other attributes as well.

Then, a one-dimensional convolution module with a
convolution kernel size of 3 is used to conduct the convolution
operation on the concatenated vector of \( \text{traj}_i \) and \( \text{attr} \) at each GPS
point \( p_i \) to obtain \( \text{loc}_i^{\text{mid}} \in \mathbb{R}^{128} \), as shown in Equation 6.

\[
\text{loc}_i^{\text{mid}} = \text{CONID}_3^{\text{elu}} \left( \text{traj}_i \circ \text{attr} \right), \tag{6}
\]

\( \text{CONID}_3^{\text{elu}} \) represents a one-dimensional convolution
module with a kernel size of 3 and the activation function of
ELU function. Thus, the resulting middle feature matrix is
\( \text{loc}_i^{\text{mid}} \in \mathbb{R}^{128 \times |T| - 2} \).

Since, in our task, the travel time is highly related to the
total distance of the path, we further append a column to the
previously obtained feature map \( \text{loc}_i^{\text{mid}} \). The i-th element of the
new appended column is Disgap \( (p_i, p_{i+2}) \), which
represents the distance gap between \( p_i \) and \( p_{i+2} \), as well as the
distance of the i-th local path. Finally, a one dimensional
convolution process with kernel size of 1 is carried out on the
concatenated vector of \( \text{loc}_i^{\text{mid}} \) and Disgap \( (p_i, p_{i+2}) \) to obtain the
vector \( \text{loc}_i^f \in \mathbb{R}^{128} \) passed into the multiview CNN
Transformer component, as shown in Equation 7. Thus, the fusion
feature map is \( \text{loc}_i^f \in \mathbb{R}^{128 \times |T| - 2} \).

\[
\text{loc}_i^f = \text{CONID}_1^{\text{elu}} \left( \text{loc}_i^{\text{mid}} \circ \text{Disgap}(p_i, p_{i+2}) \right). \tag{7}
\]

Through these two CNN modules of the path encoder
component, the local spatial features of the trajectory are
extracted and additional information is integrated.

### 3.2. Multiview CNN Transformer component

The fusion feature map \( \text{loc}_i^f \) preliminarily extracts the spatial features of the
local paths, but the temporal dependencies and the
deeper spatial dependencies need to be further captured.
Thus, we introduce the multiview CNN Transformer
component in our model. Nowadays, CNN has been the
dominant model for visual tasks, and transformer structure
is the most advanced in natural language processing tasks
[7, 8]. Using Transformers to accomplish visual tasks and
sequence data prediction has become a hot research at
present [9]. Therefore, we discuss about whether trans-
former can better learn complex patterns and dynamics
from the traffic trajectory sequence data and obtain more
accurate traffic trip time prediction by using the self-at-
tention mechanism.

Through the experiments, we found that replacing LSTM
with a traditional transformer coding layer alone did a poor
job of capturing spatial-temporal characteristics, possibly
because some important information was lost when data was
passed through multilayer transformer. Therefore, we uti-
liized to combine CNN and transformer to extract the spatio-
temporal features of the trajectory. Generally, CNN with
multiple convolutional kernels of different sizes often
achieves better results than the model with single convolutional
kernels, but the application of multiview CNN in sequence
processing is not common because the convolution network
is always accompanied by the constant change of sequence
length [10, 11]. However, we utilize the multiview CNN
Transformer component to process the spatio-temporal
trajectory data in the experiment and achieve better pre-
diction results. Thus, inspired by STTN [9], we proposed multiview CNN Transformer component. The structure of
the proposed multiview CNN Transformer component is
demonstrated in Figure 4, which consists of a series of
multiview CNN Transformer (MCT) blocks. The bottom left
of Figure 5 illustrates the structure of MCT block, and the
bottom right illustrates the structure of multiview CNN
layer. We illustrate the spatial and temporal transformer of
MCT block with one multihead attention layer and one feed-
forward layer. More specifically, a MCT block contains a
spatial transformer and a temporal transformer to jointly
learn spatial-temporal features in the context of dynamically
changing dependencies. The combination of several MCT
blocks can form a deep model for capturing more com-
licated spatial-temporal features.

Compared to STTN, which deals with the problem of
traffic flow forecasting, we improve the network structure by
adding multiview CNN module to enable the network to
extract multilevel spatial correlation information, so as to be
suitable for traffic travel time estimation problem.

### 3.2.1. Spatial Transformer

The proposed spatial transformer has three components: spatial-temporal position layer, transformer layer, and the multiview CNN layer. More specifically, the spatial-temporal position layer is learned to incorporate spatial-temporal position information of each node. The transformer layer is adopted to explore the road
topology information for hidden spatial dependency pat-
terns. The multiview CNN layer is used to capture the
hidden spatial dependencies evolving with the variety of

### 3.2.2. Transformerlayer
views, avoiding the narrow field of vision caused by too dense trajectory sampling.

Since trajectory information and external information have been embedded and fused in the traffic information fusion component, simply the position encoding layer of the common transformer is adopted here.

Transformers have been proved to have excellent performance on NLP problems, and the core of the transformer is self-attention mechanism [5]. Self-attention mechanism can well establish the connection between each node and all other nodes, so it has an excellent effect on continuous sequence processing. However, in our opinion,
the pure transformer is more suitable for capturing temporal features than spatial features. Transformer focuses more on the connection between all nodes in the whole sequence and does not pay special attention to the connection between local node sequences. For example, at two adjacent intersections, the probability of a traffic jam at one intersection increases significantly after the traffic jam at the other intersection, while the probability of a traffic jam at the intersection far away from them does not change significantly. Therefore, we need to optimize transformer to be appropriate for extracting the spatial features of trajectories.

The proposed spatial transformer component adds a multiview CNN layer to the original transformer structure, which achieves good results in the field of computer vision such as image classification and image recognition. The core of multiview CNN is to convolve the image with multiple convolution kernels of different sizes and process the image in different pooling ways after concat feature vectors. This approach has better performance than CNN with a single convolutional kernel. Multiview CNN is also applied in the field of text processing and emotion analysis. One-dimensional convolution with different convolution kernel sizes is usually used for text processing, and the obtained non-isometric vectors are gradually spliced, and the final output results are obtained after pooling in different ways.

Transformer model requires equal dimension input and output, that is, the vector of the input transformers should be consistent with the vector dimension of the transformer output. Therefore, the feature vectors of the incoming model are firstly extracted in different view domains using one-dimensional convolutional neural networks with different convolution kernel sizes, and then mapped to the target domain in a nonlinear way as follow:

$$X_{view}^{m-1} = \text{elu} \left( W_{View} \cdot X_{pos} + b_v \right), \quad i \in [1, 3],$$

where $X_{view}^{m-1}$ represents the input vector of the $m$-th MCT block after position layer, $W_{View}$ represents the one-dimensional CNN of different kernel sizes. In this paper, three one-dimensional convolutions with convolution kernel size of 1, 3 and 5 are adopted. $b_v$ is the offset vector and $loc_{view} \in \mathbb{R}^{128 \times (|T| - 2)}$ is the feature matrix of different views.

Next, we extend the external information vector $attr \in \mathbb{R}^{128}$ to the external information matrix $Attr \in \mathbb{R}^{(|T| - 2) \times 128}$ by replication $|T| - 2$ times, and then multiply it with the feature matrix of different views $loc_{view}$, and get the corresponding view matrix $View_{i} \in \mathbb{R}^{128 \times 128}$ under different views after softmax operation as follow:

$$View_{i} = \text{softmax} \left( loc_{view} \cdot Attr \right).$$

After that, the features of each node are extracted again for feature enhancement by multiplying with the feature matrix, and then summed up to obtain the local feature matrix $X_{view}^{m-1} \in \mathbb{R}^{(|T| - 2) \times 128}$ as follow:

$$X_{view}^{m-1} = \text{elu} \left( W_{MVC} \cdot \sum_{i=1}^{3} (View_{i} \cdot X_{pos}) \right).$$

$W_{MVC}$ is a learnable weight matrix. We use $\text{Transformer}_{S}(\cdot)$ to denote the encoder structure of generic transformer module in the spatial transformer component. Thus, the output feature matrix of the spatial transformer of $m$-th MCT block can be carried out finally as:

$$\text{out}_{S}^{m} = \text{Transformer}_{S}(X_{view}^{m-1}) + X_{view}^{m-1}.$$

3.2.2. Temporal Transformer. Transformer has a great ability to establish connections between nodes in a sequence by capturing the role and dependencies of each node on the others. It performs much better in sequence processing than RNN and LSTM, which are prone to problems such as gradient explosion. Thus, we also adopt a temporal transformer to efficiently and effectively capture long-range temporal dependencies over time:

$$\text{out}_{T}^{m} = \text{Transformer}_{T}(\text{out}_{S}^{m}),$$

where $\text{Transformer}_{T}(\cdot)$ represents the encoder structure of generic transformer module in the temporal transformer component. $\text{out}_{T}^{m}$ is the output feature matrix of the temporal transformer of $m$-th MCT block.

Considering that multilayer networks may cause gradient disappearance, we introduce the input before the incoming module at the tail of each module as the residual connections, and finally obtain the overall spatio-temporal feature matrix of $m$-th MCT block as follow:

$$X^{m} = \text{out}_{T}^{m} + X_{pos}^{m-1}.$$

Then, the output of $m$-th MCT block $X^{m}$ will be used as the input of the $m+1$-th MCT block until the fusion traffic information passes through all MCT blocks and finally obtains the output feature matrix $X^K$ ($K$ is the number of combined MCT blocks) of the entire multiview CNN Transformer component.

In general, the proposed multiview CNN Transformer component enhances data by adding residual structure and multiple convolution kernel CNN of different sizes to two traditional transformers to avoid losing some important information in the process of data transmission. After the traffic information fusion component, the data were firstly fused with location information, and the partial temporal features were extracted by multilayer mechanism and the spatial features were further extracted by multiview convolution module. Then the output of the attention mechanism is fused with the output of CNN, which is transmitted to a transformer encoding layer for further extraction of time features. The output vector is transmitted to the transformer of the next layer by adding the vector of the previous state transmitted to residual.

Now, by utilizing the multiview CNN Transformer component, we obtain the sequence $\{x_{1}, x_{2}, \cdots, x_{|T| - 2}\}$ ($x_{i}$ represents the $i$-th row of $X^K$) which represents the spatial-temporal features of the given GPS trajectory.
3.3. Multi-task learning component. The output sequence of multiview CNN transformer component varies with the length of each trajectory, so it is necessary to treat it as a vector of the same length in subsequent processing. Generally, the reason for the insufficient prediction time mainly comes from some local paths. For example, a local area containing multiple intersections and signal lights will easily lead to traffic congestion and increase the overall prediction time. Therefore, more attention should be paid to these paths. The attention mechanism with external information fusion adopted by DEEPPTTE model performs well in this respect [12]. Therefore, a multitask learning component which combines the previous components and estimates the travel time of the input path is carried out. Since this component directly follows the existing methods, we will not make too many statements here, just list the key formulas for completeness.

3.3.1. Local Path Estimation. Recall that we use the multiview CNN Transformer component to obtain an output feature sequence \( \{x_1, x_2, \ldots, x_{|T| - 2}\} \), and each \( x_i \) corresponds to the spatial-temporal feature of local path from \( p_i \) to \( p_{i+2} \) (since the convolution kernel size is 3). Three stacked fully connected layers are simply used here to map each \( x_i \) to a scalar \( t_i \), which represents the travel time estimation of the i-th local path.

\[
x_{\text{attention}} = \sum_{i=1}^{\lfloor |T| - 2 \rfloor} \alpha_i x_i, \tag{14}
\]

where \( \alpha_i \) is the weight for the i-th local path, and \( \sum_{i=1}^{\lfloor |T| - 2 \rfloor} \alpha_i = 1 \). We consider the spatial information of the local paths, as well as the external factors to learn the weight parameter \( \alpha_i \):

\[
\begin{align*}
\alpha_i &= \frac{e^{\sigma_{\text{attention}}(\text{attr}, x_i)}}{\sum_{j=1}^{\lfloor |T| - 2 \rfloor} e^{\sigma_{\text{attention}}(\text{attr}, x_j)}}, \\
\sigma_{\text{attention}} &= \langle \cdot, \cdot \rangle
\end{align*}
\tag{15}
\]

where \( \langle \cdot, \cdot \rangle \) represents the inner product operator and \( \sigma_{\text{attention}} \) is a nonlinear mapping which maps attr to a vector with the same length as \( x_i \).

Finally, \( x_{\text{attention}} \) is passed through several stacked residual fully connected layers and a fully connected layer to obtain the estimation of the entire path. We use \( t_{\text{entire}} \) to represent the travel time estimation of the entire path.

4. Experiment and Analysis

In this section, we report our experimental results on two large-scale real-world datasets compared with several travel time estimation methods. The contribution of each component in the model is also demonstrated by ablation experiments.

4.1. Settings

4.1.1. Data Preparation. We use two different sets of real-world traffic datasets (the same as [12]), and each dataset contains two sub-datasets: trajectories/trips and external factors, detailed as follows:

(1) Chengdu Dataset: Chengdu Dataset consists of 9,737,557 trajectories of 14864 taxis in August 2014. The shortest trajectory contains only 11 GPS records (2km) and the longest trajectory contains 128 GPS records (41km).

(2) Beijing Dataset: Beijing Dataset consists of 3.1 million trajectories of 20442 taxis in April 2015, Beijing. The shortest trajectory contains 15 GPS records (3.5km) while the longest trajectory contains 128 GPS records (50km).

According to the proportion 2:1:1, the data are divided into the historical dataset, validation dataset, and test dataset. The historical dataset is used to train the MCT-TTE model; the validation dataset is applied to parameter calibration; and the test dataset is utilized for performance evolution.

Data processing should be completed before prediction because outliers, missing values, and other unfavorable factors in raw data impact the performance of travel time estimation models [4]. Therefore, we first adopt noise filtering to remove some noise points from a trajectory that may be caused by the poor signal of location positioning systems. Then, similar to Liu [13], we set some screening conditions according to the common characteristics of traffic travel trajectory and annotate and eliminate the obvious abnormal trajectory. The screening criteria include the following: the travel distance greater than 100km or less than 0.5km, the average speed greater than 100km/h or less than 5km/h, and the travel time greater than 7200 seconds or less than 60 seconds. Finally, we carried out the Map-Matching to project each point of a trajectory onto a corresponding RS where the point was truly generated. Map matching is a process to convert a sequence of raw latitude/longitude coordinates to a sequence of road segments. Knowledge of which road a vehicle was/is on is important for assessing traffic flow, guiding the vehicle’s navigation, predicting where the vehicle is going, and detecting the most frequent travel path between an origin and a destination, etc. We used an open source tool which is called Leuven.MapMatching to align a trace of coordinates (e.g., GPS measurements) to a map of road segments based on a Hidden Markov Model (HMM).

The two trajectory data sets are different in some static features. Chengdu data set has such features as driver id and passenger status, while Beijing data set has such features as road network data and real-time road condition. The external information missing from the original dataset (such as weather and road network information, etc.) will be supplemented by reviewing historical data manually. The statistics for the two datasets (after preprocessor) are shown in Table 1.
### 4.1.2. Parameter Setting. The parameters we used in our experiments are described as follows:

1. In the RSV embedding model, we set the size of the sliding window to 3, the ratio of positive and negative samples to 1:5, and each dimension of mapping each road segment to the center/upstream/downstream vector space is 10.
2. In the path encoder module, we embed passenger status to \( \mathbb{R}^2 \), \( \text{Week}_\text{id} \) to \( \mathbb{R}^3 \), \( \text{Time}_\text{id} \) to \( \mathbb{R}^8 \), \( \text{Driver}_\text{id} \) to \( \mathbb{R}^{16} \), \( \text{Weather}_D \) to \( \mathbb{R}^4 \) and \( \text{Weather}_N \) to \( \mathbb{R}^4 \).
3. In the multiview CNN Transformer component, we use 6 MCT blocks.
4. In the multitask learning component, we use a fully connected layer with tanh activation as \( \sigma_{\text{attention}} \) in Equation 15. We use total three stacked residual fully connected layers with the input and output dimension of 128, and the ReLU function as the activation. The finally fully connected layer is from dimension 128 to 1.

### 4.1.3. Model Training. We used the mean absolute percentage error (MAPE) as our objective function during training phase, since MAPE is a relative error which can guarantee our model to provide accurate results for both the short path and long path. We use both MAPE and percentage error (MAPE) as our objective function during the training phase, since MAPE is a relative error which can guarantee our model to provide accurate results for both the short path and long path. We use both MAPE and percentage error (MAPE) as our objective function during the training phase, since MAPE is a relative error which can guarantee our model to provide accurate results for both the short path and long path.

For the local path estimation, we define the corresponding loss as the average loss of all local paths:

\[
\text{Loss}_{\text{local}} = \frac{1}{|T|-1} \sum_{i=1}^{|T|-2} \left[ t_i - \text{timegap}(p_i, p_{i+1}) + \varepsilon \right],
\]

(16)

where \( \varepsilon \) is a small constant to prevent the explosion when the denominator is close to 0, and \( \text{timegap}(p_i, p_{i+2}) \) represents the time gap between \( p_i \) and \( p_{i+2} \). For the entire path, we define the corresponding loss as:

\[
\text{Loss}_{\text{entire}} = \frac{t_{\text{entire}} - \text{timegap}(p_1, p_{|T|})}{\text{timegap}(p_1, p_{|T|})}.
\]

(17)

Our model is trained to minimize the weighted combination of two loss terms:

\[
\text{Loss} = \beta \cdot \text{Loss}_{\text{local}} + (1 - \beta) \cdot \text{Loss}_{\text{entire}},
\]

(18)

where \( \beta \) is the combination coefficients that linearly balances the tradeoff between \( \text{Loss}_{\text{local}} \) and \( \text{Loss}_{\text{entire}} \). Different combination coefficient \( \beta \) will be tested from 0 to 0.99 to determine the best setting.
the best performances are marked in bold. As we can see, our model achieves the best performance in both Chengdu and Beijing datasets. Compared with Average, TEMP and XGBoost methods, our model has a significant improvement in performance and has certain optimization compared with the recent advanced methods DeepTTE and DeepGTT.

We further compare our model with the two methods with the best performance in baseline: DeepTTE and DeepGTT on Chengdu dataset. Figure 5 shows the changes of MAPE under different travel times. We can see that:

1. The MAPE of MCT-TTE is smaller than that of DeepTTE and DeepGTT model in different travel time, which indicates that MCT-TTE has better performance in different travel time.

2. With the increase of travel time, the prediction accuracy of the three models decrease. However, compared with DeepTTE and DeepGTT, the variation of MCT-TTE model is smaller, and it is more stable, especially for long trips.

We also studied the impact of travel distance on prediction accuracy. Figure 6 shows the changes of MAE and MAPE under different travel distances. Through observation, we can see that:

1. The MAE and MAPE of MCT-TTE were smaller than that of DeepTTE and DeepGTT models at different travel distances, which indicates that MCT-TTE has better performance under different travel distances.

2. With the increase of travel distance, MAE of the three models gradually increase, while MAPE shows a decreasing trend. This indicates that with the increase of distance, the absolute error of prediction increases gradually, but the relative error decreases gradually.

### 4.3. Ablation Experiments

In order to further verify the effect of some certain components of the proposed model, we conducted several ablation experiments on MCT-TTE:

1. **with/without RSV embedding model**: The part of MCT-TTE that uses RSV embedding model to map the road sequence is removed, that is, the road network information is not used for estimation, and traj does not contain all information related to the road network.

2. **with/without multiview CNN**: Replace the multiview CNN layer with the general residual layer in multiview CNN transformer component.

3. **with/without local path estimation**: Only use the entire path estimation in multitask learning component to train and evaluate MCT-TTE.

The ablation experiment results are shown in Table 3. Results show that the three modules of RSV embedding model, multiview CNN, and local path estimation are all effective for the whole MCT-TTE model. Among them, removing the multiview CNN component has the most significant effect on the estimation accuracy, which indicates that the multiview CNN layer can capture the spatial correlation information between trajectory points more effectively. The existence of local path estimation component also effectively improves the final accuracy of the model. More importantly, we found in the training procedure that the convergence rate of the model would be greatly reduced if the local path estimation component was removed. Road network information improves the MAPE by about 2%, which is the icing on the cake.

Through experiments, we found that although the road segment vector embedding model based on Skip-Gram is trained separately, its training convergence is slow, especially when the trajectory data is concentrated on several main roads and there is little data on other road sections. Therefore, we balance the model prediction accuracy and training efficiency by adjusting the size of the sliding window. In addition, the phenomenon of slow convergence of training can be improved to some extent by discarding the road segments with very low utilization rate (equivalent to rare words in natural language processing) during map matching.

### 5. Related Work

There is a large body of literature on the estimation of travel time, we only mention a few closely related ones.

#### 5.1. Deep Learning in multivariate time series prediction

In deep learning-based methods, network structures represented by MLP, RNN, and LSTM have opened the prelude to deep learning to process time series data [17–19]. For simple univariate time series prediction, the use of LSTM can generally achieve better results. Multivariate prediction models based on deep learning are usually the standard RNN models combined with ARIMA and MLP hybrid models in the early stage. Later, Dasgupta proposed to use dynamic Boltzmann machine and RNN to jointly predict multivariable variables [20]. Subsequently, Gonzalez proposed an auto-encoder Convolutional Recurrent Neural Network to combine the Convolutional RNN with the Encoder [21]. Shih proposed the TPA-LSTM model, which processes input...
through a circular neural network and then uses a convolutional neural network to calculate attention scores across multiple step sizes [22]. Yin proposed an integrated framework for dynamic imbalanced learning to achieve a vulnerability exploitation time prediction [23].

5.2. Data driven based on vehicle history. The data-driven approach based on the vehicle’s historical trajectory refers to the use of a large number of trajectory data to learn the change relationship of trajectory points, which involves noise point removal, map matching, vehicle charging scheduling, trajectory comparison, trajectory segmentation and other directions, and then can be used for trajectory classification, travel prediction, outlier detection and other downstream tasks [4, 13]. We focus on the research progress of travel prediction here. Existing solutions for travel time prediction tasks fall into two categories. The first is a path-based solution that uses an intuitive physical model to represent travel time: the total travel time for a given route is represented as the sum of the travel time through each segment and the delay time at each intersection [24]. The second category is data-driven solutions that use location-based data to build rich features and build high-dimensional feature mappings for them. Among them, the first kind of scheme has the advantage of strong interpretability, and the travel time is the sum of the time of a single road section. However, due to the separate prediction of different sections, the accumulation of errors in multiple sections, and the influence of factors such as intersections and traffic lights, the prediction accuracy of this method is not high. Wang proposed a method based on the nearest neighbor, which estimated the travel time of the current trajectory by averaging the travel time of all historical trips with similar starting points and end points [14]. However, this nonparametric method is difficult to be generalized to the case with no neighbors or a very limited number of neighbors. In Jindal, ST-NN, a multilayer feed-forward neural network for travel time estimation first takes

| Dataset | Metrics | w/o RSV | w/o multiview CNN | w/o local estimation | MCT-TTE |
|---------|---------|---------|-------------------|----------------------|---------|
| Chengdu | MAE [s] | 178.33  | 314.11            | 253.92               | 154.76  |
|         | MAPE [%]| 13.69   | 23.56             | 18.24                | 11.25   |
| Beijing | MAE [s] | 188.97  | 357.19            | 281.76               | 164.28  |
|         | MAPE [%]| 13.48   | 24.36             | 20.14                | 11.78   |
the discrete longitude and latitude of the starting point and destination as the input to predict the travel distance, and then combines this prediction with time information to estimate the travel time [25]. Lan gave a clear classification of travel time prediction, believed that local traffic conditions were closely related to land use types and building conditions, and designed a multitask end-to-end learning framework to learn travel time [26]. Wang proposed the DeepTTE model, which combined CNN and LSTM to extract the temporal and spatial features of trajectory data, and integrated the attention mechanism to predict travel time [12]. Dai proposed AutoNavi’s latest travel time prediction model, which takes the planned traffic flow in the user’s travel intention as an approximation of the actual traffic flow in the future [27]. This method can effectively obtain the traffic characteristics, but the data of such methods need to be acquired in real time, so it is difficult to be popularized in practical application.

6. Conclusion

In this paper, we study the problem of estimating the travel time for a given trajectory. We propose an end-to-end framework based on Transformer and Convolutional Neural Network called Multiview CNN Transformer Travel Time Estimation (MCT-TTE) model. Our model can effectively capture the spatial and temporal dependencies of the given trajectory at the same time. Our model also fuses various factors which may affect the travel time, such as driver habit, travel date, weather, and most importantly the road network. Experiments on two common datasets show that the proposed MCT-TTE achieves a high estimation accuracy. At the same time, the MCT-TTE model has the ability to easily extend and integrate more additional information, which may be a flexible and efficient framework of travel time estimation problem. We will further study the problem of reduced training efficiency caused by ERSV model and Transformer structure in the future.

Data Availability

We evaluate our model on two large open scale datasets Chengdu Dataset and Beijing Dataset, which are the same with published paper [12]. We have done some data preprocessing on the original data. All of the code and the experimental data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding this work.

References

[1] K. N. Qureshi and A. H. Abdullah, “A survey on intelligent transportation systems,” Middle East Journal of enitic Research, vol. 15, no. 5, pp. 629–642, 2013.
[2] F. A. Gers, D. Eck, and J. Schmidhuber, “Applying lstm to time series predictable through time-window approached,” in Proceedings of the International Conference on Artificial Neural Networks, Spring, Vienna, Austria, August 2001.
[3] G. P. Zhang, “Time series forecasting using a hybrid arima and neural network model,” Neurocomputing, vol. 50, pp. 159–175, 2003.
[4] Y. Zheng, “Trajectory data mining,” ACM Transactions on Intelligent Systems and Technology, vol. 6, no. 3, pp. 1–41, 2015.
[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed repre-sentations of words and phrases and their compositionality,” Neural Information Processing Systems, vol. 26, pp. 3111–3119, 2013.
[6] J. Shi, J. L. Mao, and C. Q. Jin, “Travel time prediction for urban road based on spatial-temporal dependency,” Journal of Software, vol. 30, pp. 770–783, 2019.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proceedings of the International Conference on Neural Information Processing, pp. 1097–1105, Doha, Qatar, January 2012.
[8] A. Vaswani, N. Shazeer, and N. Parmar, “Attention is all you need,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6000–6010, California, USA, December 2017.
[9] M. X. Xu, W. R. Dai, C. M. Liu et al., “Spatial-temporal transformer networks for traffic flow forecasting,” Signal Processing, 2020, https://arxiv.org/abs/2001.02908.
[10] D. A. Shajahan, V. Naylor, and R. Muthuganapathy, “Roof classification from 3-D LiDAR point clouds using multiview CNN with self-attention,” IEEE Geoscience and Remote Sensing Letters, vol. 17, no. 8, pp. 1465–1469, 2020.
[11] Z. Ouyang, C. Wang, Y. Liu, and J. Niu, “Multiview CNN model for sensor fusion based vehicle detection,” in Proceedings of the Pacific Rim Conference on Multimedia, pp. 459–470, Hefei, China, September 2018.
[12] D. Wang, J. B. Zhang, and W. Cao, “When will you arrive Estimating Travel Time Based on Deep Neural Networks,” in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), Association for the Advancement of Artificial Intelligence, New Orleans, LA, USA, February 2018.
[13] W. L. Liu, Y. J. Gong, W. N. Chen, Z. Liu, H. Wang, and J. Zhang, “Coordinated charging scheduling of electric vehicles: a mixed-variable differential evolution approach,” IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 12, pp. 5094–5109, 2019.
[14] H. Wang, Z. Li, Y. H. Kuo, D. Kifer, A simple baseline for travel time estimation using large-scale trip data,” ACM Transactions on Intelligent Systems and Technology, vol. 10, no. 2, pp. 1–22, 2019.
[15] T. Chen and C. Guestrin, “Xgboost: a scalable tree boosting system,” in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794, California, USA, August 2016.
[16] X. Li, G. Cong, A. Sun, and Y. Cheng, “Learning travel time distributions with deep generative model,” in Proceedings of the The World Wide Web Conference, pp. 1017–1027, San Francisco, USA, May 2019.
[17] D. W. Ruck, S. K. Rogers, and M. Kabrisky, “Feature selection using a multilayer perception,” Journal of Neural Network Computing, vol. 2, no. 2, pp. 40–48, 1990.
[18] J. L. Elman, “Finding structure in time,” Cognitive Science, vol. 14, no. 2, pp. 179–211, 1990.
[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[20] S. Dasgupta and T. Osogami, “Nonlinear dynamic Boltzmann machines for time-series prediction,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[21] F. J. Gonzalez and M. Balajewicz, “Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems,” 2018, https://arxiv.org/abs/1808.01346.

[22] S. Y. Shin, F. K. Sun, and H. Y. Lee, “Temporal pattern attention for multivariate time series forecasting,” *Machine Learning*, vol. 108, no. 8, pp. 1421–1441, 2019.

[23] J. Yin, M. J. Tang, J. Cao, H. Wang, M. You, and Y. Lin, “Vulnerability exploitation time prediction: an integrated framework for dynamic imbalanced learning,” *World Wide Web*, vol. 25, pp. 401–423, 2021.

[24] Y. Wang, Y. Zheng, and Y. Xue, “Travel time estimation of a path using sparse trajectories,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 25–34, New york, USA, August 2014.

[25] I. Jindal, X. Chen, M. Nokleby, T. Qin, and J. Ye, “A unified neural network approach for estimating travel time and distance for a taxi trip,” 2017, https://arxiv.org/pdf/1710.04350.pdf.

[26] W. Lan, Y. Xu, and B. Zhao, “Travel time estimation without road networks: an urban morphological layout representation approach,” in *Proceedings of the IJCAI-19*, Macao, China, July 2019.

[27] R. Dai, S. Xu, Q. Gu, C. Ji, and K. Liu, “Hybrid spatio-temporal graph convolution network: improving traffic prediction with navigation data,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3074–3082, Virtual event CA, USA, August 2020.