applied sciences

MDPI

Article

Deep Wide Spatial-Temporal Based Transformer Networks Modeling for the Next Destination According to the Taxi Driver Behavior Prediction

Zain Ul Abideen [1], Heli Sun [1,*], Zhou Yang [1], Rana Zeeshan Ahmad [2], Adnan Iftekhar [3] and Amir Ali [4]

[1] Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China; zainulabideen@stu.xjtu.edu.cn (Z.U.A.); yangzhouxa@stu.xjtu.edu.cn (Z.Y.)
[2] Department of Computer Science and Technology, Xidian University, Xi'an 710071, China; zeeshan@stu.xidian.edu.cn
[3] Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430040, China; adnan@whu.edu.cn
[4] Department of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China; amir.ali@stu.xjtu.edu.cn
* Correspondence: hlsun@mail.xjtu.edu.cn

Abstract: This paper uses a neural network approach transformer of taxi driver behavior to predict the next destination with geographical factors. The problem of predicting the next destination is a well-studied application of human mobility, for reducing traffic congestion and optimizing the electronic dispatching system's performance. According to the Intelligent Transport System (ITS), this kind of task is usually modeled as a multi-class problem. We propose the novel model Deep Wide Spatial-Temporal-Based Transformer Networks (DWSTTNs). In our approach, the encoder and decoder are the transformer's primary units; with the help of Location-Based Social Networks (LBSN), we encode the geographical information based on visited semantic locations. In particular, we trained our model for the exact longitude and latitude coordinates to predict the next destination. The benefit in the real world of this kind of research is to reduce the customer waiting time for a ride and driver waiting time to pick up a customer. Taxi companies can also optimize their management to improve their company's service, while urban transport planner can use this information to better plan the urban traffic. We conducted extensive experiments on two real-word datasets, Porto and Manhattan, and the performance was improved compared to the previous models.

Keywords: intelligent transport system; geographical factors; location base social networks

check for updates

Citation: Abideen, U.I.; Sun, H.; Yang, Z.; Ahmad, R.Z.; Iftekhar, A.; Ali, A. Deep Wide Spatial-Temporal Based Transformer Networks Modeling for the Next Destination According to the Taxi Driver Behavior Prediction. *Appl. Sci.* **2021**, *11*, 17. http://dx.doi.org/10.3390/app11010017

Received: 17 November 2020
Accepted: 14 December 2020
Published: 22 December 2020

1. Introduction

The world's population is continuously making urban centers their home, and the result is that large cities are now overpopulated. About 54% of the world's population lives in urban areas, especially in metropolises where better working and entertainment opportunities are available [1]. In cities, various jobs, education and training platforms, and fast communication modes are available within a few kilometers. In cities, increasing immediate mobility creates emissions due to heavy traffic. Optimizing the available mobility system will help enhance people's lives by providing them with more mobility and a more stable environment. The purpose of this field of research is to enable transport companies to provide better service. Better service facilities reduce fuel consumption, waiting time for customers and drivers, and traffic congestion, as well as ease of mobility.

The road traffic system has experienced rapid development; in a citywide traffic management system, road traffic monitoring and predicting plays a crucial role [2]. In recent years, mobile devices have increased dramatically. Smartphones and GPS-based portable devices have led to unexpected growth in traffic trajectory data [3]. Modeling the trajectory and destination forecasting are important not only for monitoring urban traffic but also many other exciting applications, for instance, targeted advertising based on destination, Location-Based Social Networks (LBSNs) [4], intelligent traveling schedule [5], and reducing travel costs, and energy consumption optimized scheduling strategies. Recently visited locations strongly influence the next location prediction of a trajectory [6]. Similarly, a trajectory's final destination prediction is strongly connected to existing neighboring locations [2].

In the literature, past traffic data prediction problems have been extensively studied in terms of travel time, taxi demand, and traffic flow volume. In traffic data prediction, the first applied time series prediction methods and machine learning methods include Support Vector Machine (SVM) and Autoregressive Integrated Moving Average (ARIMA) and its components [7], respectively. However, the drawback of such methods is they ignore the spatial correlation data. In recent years, deep learning, which has achieved great success in natural language process [8] and computer vision [9], for traffic prediction has been widely used [10]. Many researchers combine the Convolution Neural Network (CNN) with the Recurrent Neural Network (RNN) and to find temporal [11] and spatial correlations [9]. In traffic data prediction [12], the models merging CNN and RNN and their versions are also extensively used. In these combined methods, the spatial features are extracted by CNN first, then the temporal features by RNN, and finally the forecast is made. For taxi preallocation systems, we believe the passengers' destination information is critical. Without considering the passengers' destinations distribution, the taxi preallocation systems based solely on the forecasted taxi destination demand will suffer the following issues:

- There are city management rules that limited traffic, for example Beijing's driving restriction policy [13]. Some drivers are allowed to driving in some specified regions. Suppose a region is assigned to a taxi driver where most passengers are going to a place. In that case, that place is restricted to taxi drivers. In such circumstances, he/she cannot take an order, resulting in a waste of resources.
- To carry the passengers, some taxi drivers prefer passengers going to their familiar regions. Meanwhile, some taxi drivers are unwilling to take short trip orders for a small profit. If most passengers' destination are too close to the pick-up location or the region is outside of his/her operating regions, the driver may reject those requests.
- Suppose the driver is dispatched to an area where most passengers travel to his/her unfamiliar areas to carry the passengers to their destination. In that case, the taxi drivers may spend more time, despite being guided by GPS. Such taxi driver behaviors will reduce the level of passengers' satisfaction and taxi market operating efficiency.

To address the above challenges, we propose a novel model, Deep Wide Spatial-Temporal-Based Transformer Networks (DWSTTNs). Considering long-range modeling dependencies, the transformer becomes more efficient. As compared to the previous state-of-the-art baselines, our proposed model improves the prediction performance of the next drop-off point of a taxi. We take the whole trajectory, as our model considers the individual driver's history. The driver history consists of the most recent visited points, for instance pick-up points and drop-off points. As the geographically next drop-off point of such locations, we take FourSquare to propose the semantic illustration based on LBSN. We tested our model on two real-world datasets belonging to two different cities in the USA: Manhattan and Porto. The dataset of Porto was used with the winning model of the ECML/PKDD 2015. Our model improves on the various settings by 35% compared to the winning model ECML/PKDD. We summarize our contributions as follows:

- We propose a deep spatial-temporal transformer-based model, DWSTTNs. Our model dynamically captures spatial-temporal-based long-range dependencies. In our approach, we design encoder and decoder units. Both are the primary components of the transformer. Our model improves the prediction performance of the next drop-off point of a taxi. We take the whole trajectory, as our model considers the individual driver's history. With the help of a LBSN API, with data coming from the social website FourSquare, we encode the geographical bases of visited semantic locations.

- We design a spatial-temporal-based transformer. The spatial transformer captures the time-varying spatial dependencies dynamically. The temporal transformer captures the temporal dependencies. They work with the transformer's dynamic attention mechanism with feed-forward layers to make more efficient and accurate predictions.
- We used two real-world datasets of two different cities, namely Manhattan and Porto. Our model improves the results by a large margin compared to the previous state-of-the-art baselines.

The remainder of the paper is organized as follows. Section 2 reviews the literature on the next destination problem. Section 3 presents the preliminaries and problem statement. Section 4 presents our proposed architecture with its encoder and decoder units and the spatial and temporal transformers. Section 5 presents the results, dataset description, prepossessing, baseline comparison with the previous state-of-the-art model, experimental setup, and hyperparameter setting. Section 6 is the discussion. Section 7 concludes with future work.

2. Literature Review

Traffic forecasts have been widely studied for many decades. However, it is still an inspiring problem because of the dynamic spatial and nonlinear temporal dependencies [9,10]. In general, the research phase has gone through two stages of growth, i.e., the data-driven stage and the mathematical model-driven stage. The frameworks are delineated unevenly into parameterized, non-parameterized, and hybrid models of integration [14].

In the older phase, forecasting destination flow was predominantly driven by mathematical models, and a parametric model was often the central element of the prediction model. The standard frameworks are time series-based statistical analysis methods, for example the Autoregressive Integrated Moving Average (ARIMA) model and its variants [15]. The ARIMA model takes into account the intrinsic significance over terms of traffic variables and incorporates self-regression for prediction [16–23]. There is no need to acknowledge certain exogenous variables for their simplicity in nature and it is only driven by endogenous variables, but the time series or its distinction should be consistent. As a linear relationship model, the ARIMA model is not sufficient for explaining the association around nonlinear variables and dynamic changes.

Non-parameterized models primarily include Support Vector Regression (SVR), k-Nearest Neighbor (KNN), Support Vector Machine (SVM), and most popular Artificial Neural Networks (ANN) [24]. The models belonging to ANN, particularly deep neural networks that have been progressively established in recent decades, CNN and RNN, may define complex nonlinear relationships between variables. Ren et al. [25] proposed an efficient method for the forecasting of the nonlinear and non-periodic speed of traffic flow on city roads. This method is based on a deep convolutional strategy, for extremely fluctuating road segments. Neural networks achieved an average increase in prediction performance from 0.9% to 3.3% and improved consistently up to 23.8% for individual road segments [26]. The developed deep network architecture [27] Stacked Autoencoder (SAE) for traffic local highway network flow regression suggests that deep networks could better capture the features of data spatial-temporal correlation compared to traditional shallow models, such as machine learning algorithms, backpropagation neural networks, and SVM, for prediction. It could be possible to increase accuracy [28] substantially. Since CNN is able to retrieve the correlation of spatial characteristics and RNN can obtain the correlation of temporal features, the incorporation of the two has the ability to distinguish visual characteristics. It is also commonly used for the spatial-temporal study of traffic flow. Duan et al. [29] designed Spatial Recurrent Convolutionary Networks (SRCNs) to transform motor vehicle speeds in urban road networks into images, estimated vehicle speed using image sequence analysis, and evaluated its greater efficiency compared to models such as SVM and SAE [30]. The authors of [31] used a deep neural hybrid network made up of CNN and a Long Short-Term Memory (LSTM) to study the GPS taxi trajectory data and predict the city-wide road network.

Taking into consideration the spatial-temporal features of the traffic flow, the adjustments have expanded the conventional forecast of traffic movement from road segments to the whole city-wide road network [32]. In addition, Ma et al. [33] implemented RNN to the taxi destination forecast, predicting the destination of the taxi on the basis of the GPS location of the taxi after its departure. The results indicate RNN's excellent forecast performance [34]. Deep learning technologies have been used to forecast the future of a recurrent neural network-restricted Boltzmann machine deep correlation model formulated using congestion in large-scale transport networks. Yao et al [35] implemented GPS trajectory information to forecast the formation, distribution, and dissipation of traffic congestion [36]. To take more exogenous dependencies into account, the deep fusion network [37] and the Deep Spatial-Temporal Multi-View Network (DMVST-Net) [38] were developed to determine the demands of individuals for online taxi-hailing applications, which outperformed ARIMA and multi-layer perceptron. The authors of [39] presented the diffusion of a convolutionary recurrent neural network for traffic flow forecasting and attained a 12–15% rise in the forecast's performance [40].

In 2008, RNN technology was implemented to analyze wireless communication network traffic, which was the first attempt to use deep learning-based knowledge for network destination projection [41]. RNN performed better than the Gravity method when forecasting destination problem, demonstrating the vast potential of RNN in the network analytics. Much research has been performed on the current stage of the taxi destination flow forecast, and various traffic data have been used in this research. Hlupić, et al. [42] utilized LSTM to collect data flow of metro travelers in order to predict complex destination flows among metro network locations, which enhanced performance compared to conventional VAR and Calendar methods [43]. Nevertheless, this technique cannot fully forecast the dynamic taxi destination flows because the metro lines are comparatively fixed, the roads or highways have a small scale compared to the metropolitan road traffic network, and the time between the metro stations is static. In particular, as traffic data have grown more widely, non-parameterized forecast models have become the more accurate traffic forecasting models. They have started to play a role in nonlinear, static, and dynamic processes and spatial-temporal analyses. In addition, these kinds of models can be coupled with the environmental influences for further enhancing prediction accuracy [44].

To address the limitations of the methods discussed above, we propose the novel DWSTTNs model. In our framework, the encoder and decoder are the primary units of the transformer. The Location Based Social Networks (LBSN) application helps; we encode the geographical information bases od visited semantic locations. In particular, we trained our model for the exact longitude and latitude coordinates to predict the next destination.

3. Preliminaries

In this section, we discuss the formal definition of the taxi destination problem. We also provide the important notations and previous research theory. Then, we briefly introduce the transformer with its encoder and decoder with an attention mechanism. In the end, we provide the implication of our proposed solution.

Definition 1. *Let us have a spatial-temporal point ω. The expression ω is the couple of ω=(t, L) visited, where t is a time interval and the longitude and latitude are the location $L = (i, j)$. At the visited location, i and j are the spatial coordinates. These coordinates are given in a Coordinate Reference System (CRS), namely the longitude and latitude, CRS World Geodetic System (WGS84).*

Definition 2. *We predict the taxi's next destination according to the driver's behavior. Let β be a taxi driver, the trajectory T_β is a path where the taxi driver rides the taxi, T_β= $P_1, P_2, \ldots, P_n$. The trajectory time-ordered sequence is composed of the alternative passenger pick-up and drop-off points based on spatial-temporal facts. We provide the notation of the last rides of the taxi driver β as $\frac{n}{2}$. In Figure 1, we illustrate Definitions 1 and 2.*

Problem Statement. As per definitions, for the taxi driver β, the trajectory of the taxi is T_β. $T_\beta \cup P_{n+1}$, where P_{n+1}, is the current pick-up point. Our problem is finding the taxi's next destination according to the behavior of the taxi driver. The next destination prediction is the drop-off location $l_{n+2} = (i_{n+2}, j_{n+2})$, which is the actual destination of the taxi driver β and $l_{n+2} \in \mathbb{R}^{ixj}$.

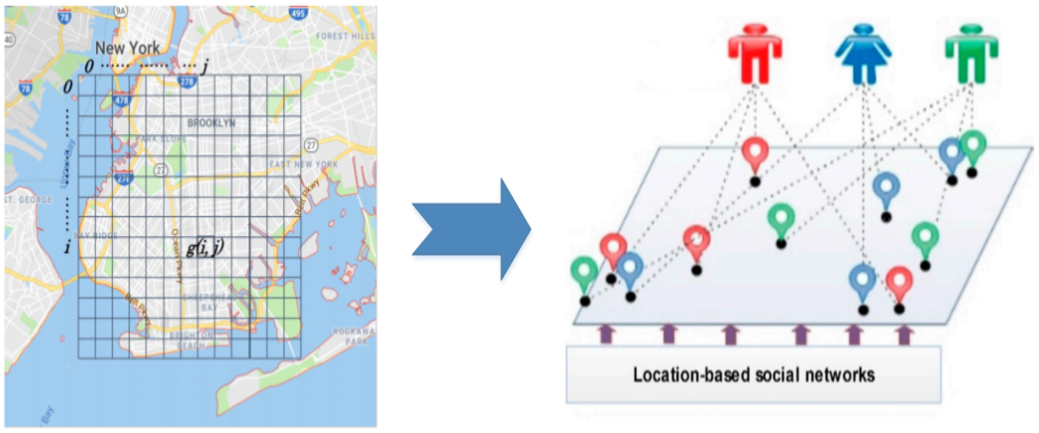

Figure 1. (**Left**) The whole city graph divided into the CRS i.e., latitude and longitude; and (**Right**) the LBSN-based driver trajectory as customer pick-up and drop-off. Shaded lines show the driver trajectory, while blue, red, and green nodes show the pick-up/drop-off points.

4. Material and Method

A transformer is an architecture transforming one sequence into another. All of these processes are done with the help of the encoder and decoder. However, as compared to the previously existing sequence-to-sequence models, the transformer is different from RNN, and it belongs to the non-RNN family. The attention mechanisms are the entire backbone of the transformer, which makes it possible for the target access any part of the sequences regardless of its distance [45,46]. In essence, the structure of the transformer is organized into encoder and decoder modules, as shown in Figure 2. Similar encoder modules are stacked at the bottom of the stacked decoder modules. The construction of each encoder module is composed of a position-wise feed-forward layer and self-attention layer, while the decoder module has one additional layer as compared to the encoder part (the encoder–decoder attention layer). The encoder and decoder parts are bridged by a layer inserted between the self-attention and feed-forward layers.

During transformation, we embed trajectory sequences of each driver similar to sequences of words/tokens. Inside the transformer, the multi-head attention layers in a sequence from multiple aspects (heads) attach different importance to the words/token. The output belongs to different heads to aggregate all the information that is concatenated and passed through a linear transformation. The encoder–decoder attention layers and self-attention layers are the same as the attention mechanism behind the scenes [47]. As an example, we take self-attention. On the sequences of token as trajectories, it operates as $T = (P_1, P_2, \ldots, P_n)$, each of which is prepared by a random vector and passed through a linear transformation after being updated by using a weighted sum of any other words. The weights are assigned by their similarities, which is also called the attention score. We take the update of T_i as an example.

$$y_i = \Sigma_{j=1}^{n} a_{ij}(T_j W_V) \tag{1}$$

In the above equation, y_i is the updated of T_i and a_{ij} is the attention score. We can measure the similarity between T_i and T_j as:

$$a_{ij} = \frac{exp(C_{ij})}{\Sigma_{K=1}^{n} exp(C_{ik})} \tag{2}$$

where we measure the compatibility C_{ij} of two linearly transformed T_i and T_j. We calculate the scaled dot product as:

$$C_{ij} = \frac{(T_i W_Q)(T_j W_K)^T}{\sqrt{h}} \tag{3}$$

To strengthen the expressiveness of the transformer, three linear transformation matrices are represented as W_V, W_K, and W_Q, with h the dimension of the output. The three linear transformation matrices are equal to W.

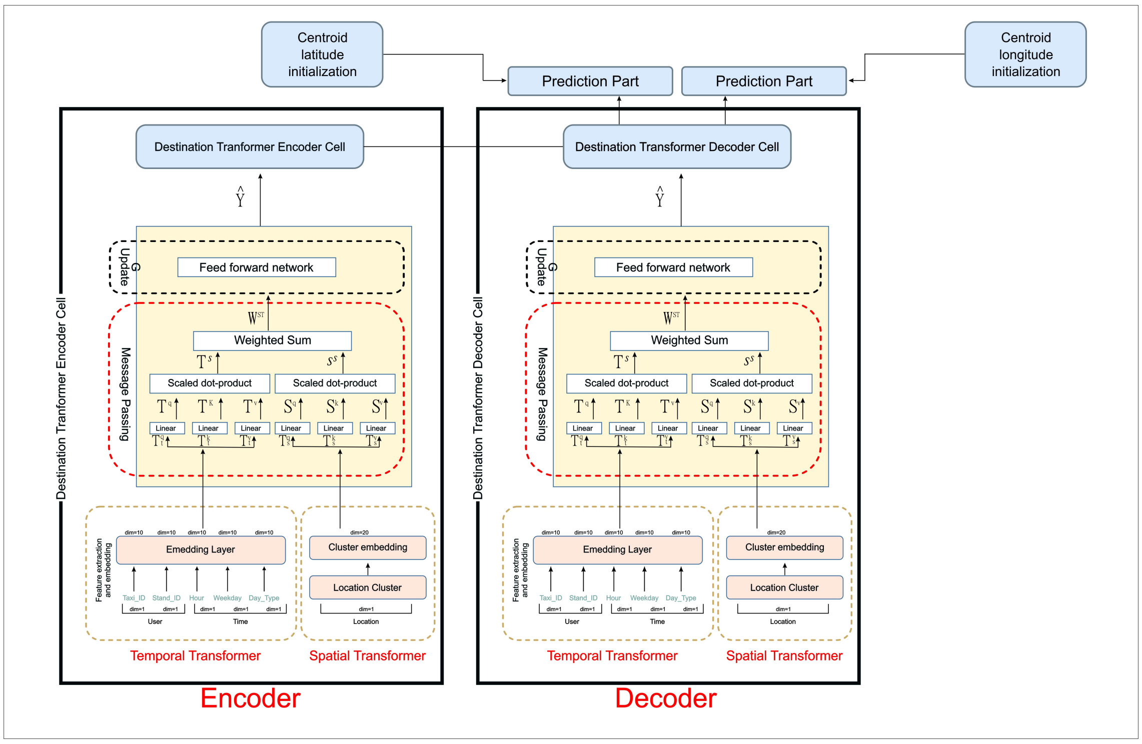

Figure 2. Main framework: (**Left**) the encoder part; and (**Right**) the decoder part.

There are five components of our deep wide model DWSTTNs, where each component works as follows.

1. We designed novel spatial and temporal transformers. In our approach, both transformers are based on a multi-modal embedding module to embed the spatial-temporal features jointly. As the encoder part, spatial and temporal transformer embedding layers convert the input token to the dimensional vectors. The input element, through the spatial and temporal transformer, make the information ordered through the positional encoding part. The purpose of positional embedding is to preserve the sequential order in a sequence. We can define mathematically positional encoding as follows.

$$pos_{embedding_{(pos,2i)}} = sin(\frac{pos}{10000^{\frac{2i}{h}}}) \tag{4}$$

$$pos_{embedding_{(pos,2i+1)}} = cos(\frac{pos}{10000^{\frac{2i}{h}}}) \tag{5}$$

In the above equation, i is the dimension of the position embedding and h is the size of the hidden layers.

$$W \bigodot pos\_embedding_{pos} \in \mathbb{R}^h \tag{6}$$

where W is the three linear transformation matrices and $\bigodot$ is the concatenation operator.

2. To deal with GPS data, we use novel sources of information to embed spatial and temporal features. The novel sources are data from LBSNs and mobile phones. These sources can be exploited for human mobility behavior data and to make city traffic more efficient. We used FourSquare. This LBSN provides the number and types of activities in the target area, also giving insight into how many people can get coverage in that zone. All the information gathered can then be used to infer the destinations of the taxi.
3. We process the sequential data belonging to GPS traces and spatial-temporal input features gathering and applied the attention mechanism. The working of the attention module cell is presented in Figure 3.
4. The transformer network learns the spatial and temporal features. Our networks' learning is based on the embedding of spatial-temporal features extracted from taxi driver behavior and semantic features.
5. Prediction is the last part of our network architecture. The encoder and decoder stack combine the output of previous modules and complete the prediction task. The prediction component consists of the SoftMax layer and the linear layer. The SoftMax layer has several neurons. The number of neurons is represented as $m = |C|$. With the help of a clustering algorithm, we define a set of location clusters C. The clustering algorithm, i.e., K-mean, trains all the trajectories of the destination. We use latitude and longitude values; each point of the trajectory is assigned to the closest centroid C_i. The coordinates of neuron clusters represent longitude and latitude by adding the two neurons with the output layer. Note that adjusting the initialized weights of the matrix with cluster centers operation is equal to the superficial linear output layer.

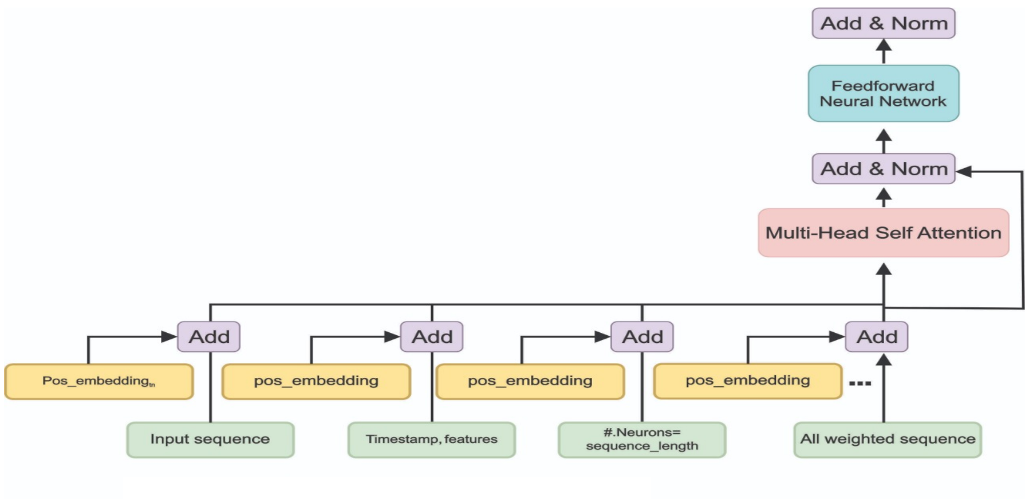

Figure 3. The working of the encoder cell.

4.1. Spatial Transformer

In this subsection, we elaborate on the component of our transformer network. The spatial transformer consists of the spatial positional embedding layer—the longitude and latitude of the pick-up and drop-off points. The spatial transformer block is fused with the liner and scaled dot product of the weighted sum. The spatial-temporal embedding layer learns to incorporate the spatial-temporal positional information, for instance location, clusters, neurons, etc. Therefore, in the transformer feed-forward structures, there are no convolutional or recurrent operations. The spatial position of a trajectory, distance, and time determine the spatial dependencies of the previous trajectory rides that should be utilized to predict the future. As the next taxi destination prediction, the transformer injects the prior positional embedding as input sequences. For the learning of our network, we adopt the learnable spatial embedding positions in the embedding layer.

Updating the grid, the input features are projected into the high-dimensionally latent subspace. These subspaces are learnable mappings. The learnable parameters are realized in the feed-forward neural networks. In the single head attention, three subspaces are obtained for each trajectory. The names of the three subspaces are (Q, K, V). In our model, we feed the query layer and the value. The linear layer gets the value through the vector. In our problem, we estimate the error rate in the distance in terms of kilometers, queries, and keys to undergo the scaled dot product to produce a score of the error in kilometers. The mathematical equations are formulated as follows:

$$[S^Q = P^S T_s^q, S^K = P^S T_s^k, S^V = P^S T_s^v] \tag{7}$$

where T_s^q, T_s^k, T_s^v are the weighted matrices of spatial transformer and S^Q, S^K, S^V are the spatial subspaces. P^S represents the spatial input features. The spatial scaled dot product of the three latent high-dimensional subspace is

$$S^S = SoftMax(\frac{S^Q(S^K)^T}{\sqrt{d_S^K}}) \times S^V \tag{8}$$

The spatial weighted matrices and subspaces are $T_s^q.S^Q, T_s^k.S^K, T_s^v.S^V = W^S$.

4.2. Temporal Transformer

In the task related to transportation, temporal dependencies are the matter for future prediction. Therefore, we develop a temporal transformer. The purpose of this is to capture the long-range temporal dependencies over time. The temporal transformer performs this task efficiently and effectively. Compared with RNN and its variants, the transformer captures the long-range temporal dependencies, and the calculations allow work with parallelization for long, scaled sequence tasks. In the temporal transformer, a similar temporal position is adopted as in the spatial transformer. The features are normalized and transferred to the attention layer. We concatenate the spatial transformer's input features in each time step with the temporal transformer positional embedding. The purpose of concatenation is to initialize the one-hot encoding and update it during the training to make it more efficient. The attention mechanism of the transformer is multi-head attention, which is important for modeling the temporal dependencies. The emporal transformer is the same as the spatial transformer computed with three high subspace temporal dependencies. The names of the three subspaces are (Q, K, V). The mathematical equations are formulated as:

$$[T^Q = P^T T_t^q, T^K = P^T T_t^k, T^V = P^T T_t^v] \tag{9}$$

where T_t^q, T_t^k, T_t^v are the weighted matrices of temporal transformer and T^Q, T^K, T^V are the temporal subspaces. P^T represents the temporal input features. The temporal scaled dot product of the three latent high-dimensional subspace is

$$T^S = SoftMax(\frac{T^Q(T^K)^T}{\sqrt{d_S^K}}) \times T^V \tag{10}$$

The temporal weighted matrices and subspaces are $T_t^q.T^Q, T_t^k.T^K, T_t^v.T^V = W^T$. The spatial-temporal weighted are $W^S.W^T \longrightarrow W^{ST} \in \mathbb{R}^h$.

4.3. Transformer Attention Module

We process the sequential data belonging to GPS traces using transformer networks. For this reason, we use the transformer encoder and decoder parts as a basic unit. ReLu is used as the activation function. For spatial-temporal input features gathering, we apply the attention mechanism. When the multi-heads finish their job, the job vector's output is added to the original positional input embedding. This is called residual connection. The residual link completes their task, producing the output. This output goes through the normalization layer. In the further processing, residual normalized production is projected towards the point-wise feed-forward network. The main advantage of the residual connections is that they train the networks directly. In other words, they allow the gradient to flow through the network instantly. The purpose of using the normalization layer is to stabilize the network. The advantage of stabilizing the network is that model training time is greatly reduced. To make the attention output richer, the point-wise feed-forward neural network is projected to the attention layer. After this task, the encoder layer ends. All operations are encoded as input; this input represents continuous attention information. This constant attention information helps the decoder; thus, the decoder focuses on the appropriate words. During the training phase, this input helps the decoder. For further information, we stack the encoder N times. The N times stack provides the opportunity for the attention layer to better learn to represent differently. This increases the predictive power of the transformer.

The feed-forward neural network is a combination of two linear layers. The ReLu activation functions are in between two layers. Two linear layers produce the output; these layers are again added to the feed-forward neural networks and further normalized. The more structural detail of the transformer is presented in Figure 3. The purpose of the attention layer is to analyze which part is more important in the trajectory and capture it. In the end, this layer is combined with the input sequence. Thus, the transformer layers capture part of the sequence, which helps the current trajectory to trace mobility regularities.

4.4. Learning Transformer Network

The temporal order data of GPS locations are based on tracing mobility. As in the previous studies, RNN outperforms different architectures such as MLPs for mobility tracing with sequential data. These networks are not tailored to work with sequential data and temporal data. However, the limitation of RNN is that the training data cannot be parallelized. The transformer network takes place with parallelization to process the sequential data and temporal data.

Furthermore, the previous research mainly focuses on the fine-grained single trajectories to predict the next taxi destination. In other words, these methods are based on all GPS location points of each ride [9]. The previous research has some important limitations; for instance, a vast volume of data of all GPS points of a trip are stored and almost all trajectory points are needed to predict the ending point. To avoid the problem mentioned above, we model the trajectory according to the definitions in the Preliminaries Section. The GPS point sequences of pick-up and drop-off are composed of pairs for several taxi rides. This is the best way to not keep all pick-up and drop-off points of the whole trajectory. In this way, the prediction takes place instantaneously; only the trip starts and pick-up location information is available. Therefore, most mobility models are set as the classification problem to predict the next location—the mobility model's goal is to classify the location we predict. We find the main drawback of this approach is during the model training, where most of the locations are unseen. The model never produces several locations. To overcome this drawback, we propose making the coordinates two different functions, longitude and latitude, to predict the destination.

The spatial part has the location clusters. In the prediction part, as mentioned above, we define the set of clusters of locations as $m=|C|$. We calculate the clusters with the K-means clustering algorithm. The algorithm trains the destination of all the trajectories.

We keep each point of the trajectory during the training process as consequently assigned to the nearest centroid C_i. According to the Preliminaries Section, trajectory T_β is translated into the new mapped cluster. The new clusters are traced as $CT_\beta = C_1, C_2, \ldots, C_n$. In the traced clusters, C_i is the hash of the cluster; this cluster is closest to the point. Since longitude and latitude are the points, we work with longitude and latitude values. We use the Haversine distance between the two points P_1 and P_2. The Haversine distance formula is helpful to compute the next taxi destination prediction. The formula is defined as follows:

$$D_{haversine}(P_1, P_2) = 2R\sqrt{\frac{a(dP_1, dP_2)}{a(dP_1, dP_2) - 1}} \tag{11}$$

$$a(P_1, P_2) = sin^2(\frac{(\phi_2 - \phi_1)}{2}) + cos(\phi_1)cos(\phi_2)sin^2(\frac{\lambda_2 - \lambda_1}{2}) \tag{12}$$

In the above equation, R is the radius of the Earth, (λ_i, ϕ_i). $i = 1, 2, 3...$, are the values of longitude and latitude coordinates with probability P_i.

Our model, the spatial transformer, is an embedded layer that captures all information, for instance, time, day, mobility, influence by human mobility, and semantic classification of each place. Furthermore, each location represents a single feature; we apply it to the spatial transformer cluster embedding layer as input feature sequences. In this way, the origin and destination co-occurrence is based on dense representation and then based on the other transformer module. These modules help us gather all sequential information visited in the last locations. We apply our transformer network as a basic model, encoder, and decoder to make effective human mobility for this task. Inside the transformer, the attention layer consists of multi-head attention, which passes the information as an input sequence to the feed-forward neural layer. The feed-forward layer is equivalent to applying two 1×1 convolution layers that capture the driver mobility from the previously visited locations. The final prediction part of our model consists of the SoftMax layer. This is followed by the linear layer and scale dot product layer. These layers work together to estimate the longitude and latitude values of the taxi's next destination.

4.4.1. Spatial-Temporal Feature Extraction and Embedding

Mobility in the ITS has multiple governing factors, for instance geographical and daytime characteristics of visited places. Each zone has specific activities; we assign the location of each zone a particular meaning related to the activity. Thus, the representation of a particular area is important for analyzing mobility. The benefit of this is that the information is enriched related to each location. In our model, we propose the multi-module embedding module. The module joins the spatial-temporal information and driver features to represent a vector, for the feed-forward neural networks. Some of the external features are available to make predictions more accurate; for example, weather condition features are concatenated as the input vector.

4.4.2. Taxi Driver Behavior Features

For the taxi driver's behavioral feature mapping, we map the categorical features of the taxi driver. The first important item is time; we express the time of day in hours, e.g., $h \in [1, 24]$. The total number of days in a week is represented as weekdays, $wkd = [1, 7]$. On weekdays, days are divided into different categories, such as working day, weekend, holiday, and pre-holiday. The modeling of these features is done as a one-hot representation. We make a separate spatial and temporal transformers, which produce the vector's dense representation with the embedding layer in our model network. In our model, weights are updated during the training phase.

4.4.3. Spatial Embedding with Semantic Features

The spatial semantic features are given to the set of location cluster C, and each cluster is set with the POI. Each POI assigns the nearest cluster. The details of the POI are explained in the Dataset Description Section. We map trajectory with each given point CT_β. The mapped trajectory builds the features, associated with POI. The venue of the POI is hierarchically characterized, e.g. educational places and other places to libraries and universities. The categories of POI represent the aggregated representation of the area. In the embedding portion in the spatial transformer, we model the spatial semantic features. The semantic features are based on the POI; they are associated with the trajectory. We count the macro-categories of the POI aggregated with the trajectory. The cluster embedding has the feature of measuring each category in the cluster.

4.4.4. Spatial Zone Embedding

Spatial features reflect, in a semantic manner, the time-independent features, not capturing the region's dynamics in a specific area. In this perspective, human mobility data are the best source to learn for densely representing human mobility in a city. As a result, the urban zone has similar embedding features and presents the same geometrical results as each other. In this perspective, the human mobility is seen as a language sequence. In this way, the location sequence is the sequence of words. This shows that the sequence is modeled for the textual data. The sequences model uses natural language processing. In our approach, in the hidden state, it encodes the whole sentence; each word passed to the decoder unit has a corresponding hidden state. The hidden states perform the task at each step to decode the output coming from the encoding unit. The transformer is used to translate in our approach; a word given to the positional embedding is learned from the nearby occurrence word window. The similar words in the semantic perspective have the same vector representation. Each location cluster is labeled with a textual label. In this way, we map the trajectory with the word sequences. The encoder and decoder units enable us to learn the zone embedding. The zone embedding relies on the different zones with different mobility relations. In Figure 4, the zone embedding each location cluster is labeled with a textual label.

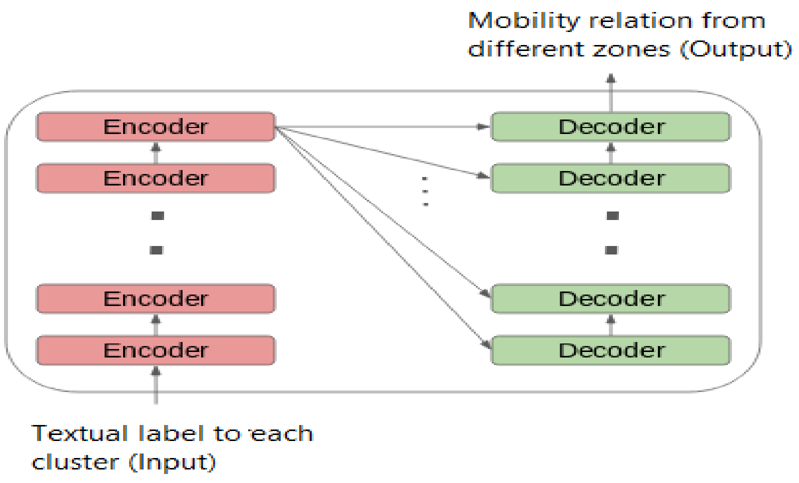
Mobility relation from different zones (Output)
Encoder
Encoder
Encoder
Encoder
Decoder
Decoder
Decoder
Decoder
Textual label to each cluster (Input)

Figure 4. Encoder and decoder associations with cluster labeling.

4.5. Prediction Component

The prediction part is the last part of our network architecture. The encoder and decoder parts complete the task to transfer tasks through feed-forward neural networks towards the prediction part. The job of the decoder is to produce the output sequences receiving input from the encoder. The decoder has the same sublayers as the encoder. The decoder part consists of multi-head attention layers, residual connections, normalization layers, and the feed-forward neural networks layer after each sublayer. All of these layers work the same as compared to the encoder part; the only difference is that each multi-head attention layer works differently. Inside the decoder, SoftMax gets the word probabilities.

The decoder is attached to the linear layer, acting as the classifier. The decoder begins to work with a token. The encoder takes the previous output as input; the last output is based on the encoder. The output contains the attention information. When the decoder produces output as the token, the decoder stops working. This component of our network combines the output to complete the prediction task. The prediction component consists of the SoftMax layer and the liner layer. The SoftMax layer has several neurons. The number of neurons is represented as $m = |C|$. The temporal transformer generates the number of neurons as input. The transformer network must evaluate the destination point coordinates. The coordinates are represented as longitude and latitude by adding the two neurons with the additional output layer. It is worth mentioning here that it adjusts the weight of the initialized matrix with the cluster equal to the simple linear output layer. The mathematical expression is defined as follows. Let the predicted destination be $\hat{y}$. The SoftMax probability layer is Pl_i.

$$\hat{y} = \Sigma_{i=1}^{C} Pl_i C_i \tag{13}$$

$$Pl_i = \frac{exp(e_i)}{\Sigma_{j=1}^{C} exp(e_j)} \tag{14}$$

5. Results

Taxi trajectory data have great interest for making better transportation infrastructures and policies—the trajectory data are used for observation, evaluation, and optimization. The modern big cities have a common problem with traffic congestion. This problem is due to the improper planning of roads, less control, and less maintenance. Our work used two real-world datasets belonging to the New York cities of Manhattan and Porto. The Porto dataset was released by the Kaggle competition ECML/PKDD 2015. The availability of this dataset allows us to compare our model with the state-of-the-art model, namely the winning approach of 2015. The taxi mobility data become enriched due to the information provided about people's activities in each visited location. The people's activities enrich the dataset with geo-located text. The features belong to POI; we used FourSquare to extract the POI features, providing each location representation.

5.1. Taxi Mobility on Porto and Manhattan Datasets

The Porto city dataset is composed of 1.7 million taxi trajectory records [48]. The dataset comes from 442 taxies. We obtained the 200,000 taxi trajectory trips. We selected 600 drivers out of the initial 5000. The dataset period is from 1 July 2013 to 30 June 2014. Figure 5 represents the pick-up and drop-off points of the taxi driver trajectories. The GPS point of each ride provides mobility trace as pick-up and drop-off locations. At the pick-up point and drop-off points, we sampled spatial-temporal features every 15 s. The Manhattan dataset is very big [49]. The Manhattan dataset was collected from 3 January 2013 to 3 January 2014. The dataset comes from 13,426 taxies, driven by 35,000 different drivers. The GPS point of each ride provides mobility trace as pick-up and drop-off locations. In Figure 6, green bubbles illustrate the pick-up and red bubbles the drop-off of the passenger. We obtained 9,100,000 taxi trajectory trips. At the pick-up point and drop-off point, we sampled spatial-temporal features every 10 s.

Furthermore, each trip has some metadata information, such as taxi id and the origin type. We take the origin of the call; day type when call is generated (working day, holiday, and weekend), and the kind of origin. When the ride starts, we attached the starting timestamp to the ride. All of the above metadata helped us collect some useful information about peoples' mobility, such as going to the office, returning home, etc. According to the Preliminaries Section, we mapped our Porto dataset according to each GPS trace. The behavior of taxi driver traces was used as an input sequence of pick-up points and drop-off points. In addition, we grouped the taxi drivers with driver id. The drivers have different timestamps; we sorted them in ascending order. The taxi pick-up points and drop-off points helped create a taxi trajectory. We made a taxi trajectory according to our trajectory definition and problem statement. In our work, we imposed $T_\beta = 8$, namely the pick-up and drop off points were made with the four past trips. Keeping the relevant history represents an excellent trade-off. In this way, to learn our model of driver behaviors, we maintained adequate trajectories for each driver. In our model, we picked the driver relative to the same time-shift driver, and we used the sequences of the trip that are not separated by more than 3 h from each other.

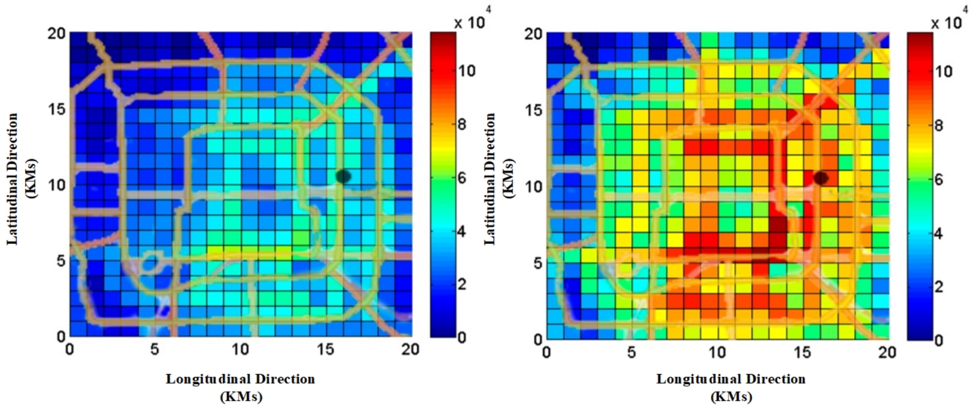

Figure 5. Taxi driver ride trajectories in KMs.

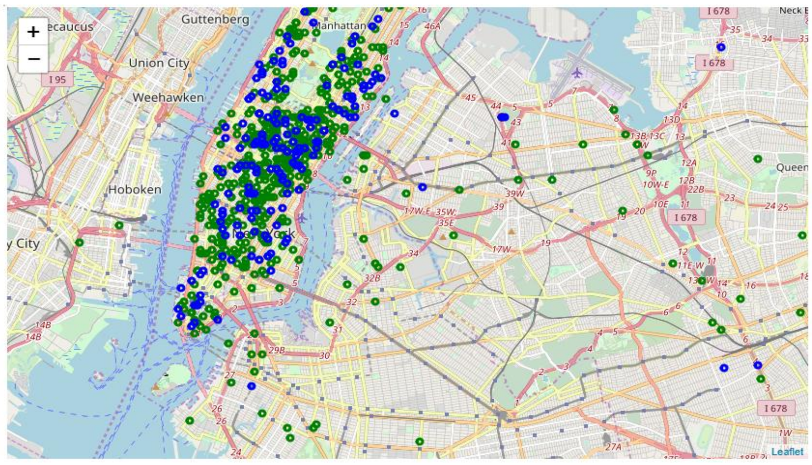

Figure 6. Blue and green bubbles illustrate the pick-up and drop-off points.

5.2. *Taxi Trajectory with Point of Interest*

The POI is usually composed of the location coordinates such as longitude and latitude. The POI also has some textual information according to the activity of that place. The particular area of POI provides a different level of details regarding activity characterization, based on the hierarchical categorization. Therefore, hierarchical categorization examples are Medical, Food, Chinese Restaurant, and Western Restaurant. In the extraction of the POI points, we used FourSquare as a source. FourSquare is a web-based geolocation social network system. This social network recommends the place with activity details. This social network API supports 100,000 requests free of charge per day. Each LBSN sets the location hierarchy according to their category. The category is based on the activity of each location in the database, such as shops or restaurants. LBSNs, following the semantic characteristics, location, and activity recommended semantically, are associated with the historical path on behalf of the POI. For example, to establish the way of activity, the Liberty Hospital path belonging to Medical is created as: Medical—Kingadward Hospital—Youhana Hospital. As we target the POI name, the structure mentioned above is more informative. The structure we established follows the combination of the features; each node keeps enriched proximity information.

LBSNs such as FourSquare consider the macro-categories of POI. The category is Professional and other places, namely Residence, College and University, shops, Food, Services, Art and Entertainment, and Travel and Transport. We extracted 8500 POI for Porto from the FourSquare source, and 65,300 POI for Manhattan. The interesting factors are census data and land usage. However, we limit our model only to POI. In addition to POI extractions, there is also the availability of other interesting information on geographical data. However, additional information is not consistently available. The other cities' dataset availability is not uniform in terms of their definition convergence. In Figure 7, we represent the historical pick-up and drop-off points to predict the next destination and repeat the input features with timestamp t.

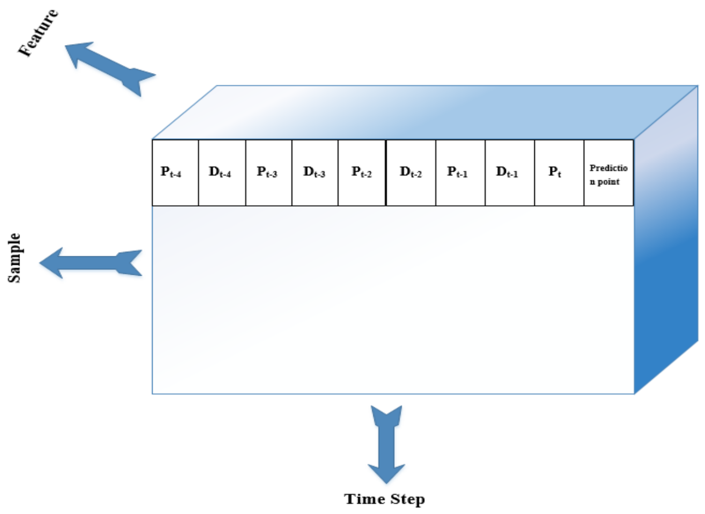

Figure 7. Driver's trajectory fed into our model (P_t are pick-up points and D_t are drop-off points).

5.3. Baselines

To demonstrate the effectiveness of our model, we compared our DWSTTNs with seven baseline methods and tuned the parameters for all models. Our model was compared with the following state-of-the-art baselines.

ARIMA. In the time series forecasting problem, the ARIMA model is a well-known model to predict the future values of time series.

Nearest Neighbors (NN). In this model, the pick-up point of the taxi is given. This model processes and generates the output. The outputs of NN are the longitude and latitude of the closest cluster centroid.

Multi-Layer Perception (MLP). This model is implemented with the winning challenge ECML/PKDD 2015. In this model, the input features are extracted. The extracted features receive the representation of the taxi trajectory. The trajectory is composed of the first five GPS points and the last five GPS points. This combination of the GPS points is associated with metadata. The MLP network contains standard hidden layers as well as 500 ReLu activation functions [41]. The MLP network follows the SoftMax layer with neurons m= |C|. If we compare our approach with the MLP approach, we assign each longitude and latitude to the given set of the cluster centroid. Thus, the output through the predicted layer is $\hat{y}$, which is the weighted average between the destination cluster centroid and the SoftMax layer. In the MLP, the problem is optimized with multi-class classification. The loss function is used for evaluation as cross-entropy. The Stochastic Gradient Descent trains the MLP model. The batch size is 200 and the learning rate is 0.001.

Multi-layer perception-SEQ (MMLP-SEQ). The MMLP-SEQ is trained with the same MLP. The only difference is not using the first five GPS points and the last five GPS points as input.

Fully Connected (FC-LSTM). The LSTM is the most straightforward traditional approach. LSTM uses only the input representation constituted by the driver and time representation. The longitude and latitude coordinates of the zone are fully connected and fed as the location in the embedding layer. The weights are adjusted randomly. The training process is allowed to update the weights.

Long- Short-Term Memory (BOC+W2V). Bag of Concept + W2V works with LSTM. The purpose of the W2V is to represent the recurrent neural network included with both BOC and W2V features. One more difference is that, instead of the zone embedding with BOC, the coordinates of the zone are embedded with W2V.

Spatio-Temporal Graph Convolutional Networks (ST-GCN). ST-GCN purely depends on the spatial and temporal components. ST-GCN uses graph weights with spatial-temporal features to embed with transformer architecture. The traffic transformer captures the continuity and periodicity of time series and models the spatial dependency.

5.4. Experimental Setting

In our experimental settings, all experiments were conducted on the NVIDIA 2060 RTX GPU and Google clouds. We trained our proposed model with a Mean Squared Error (MSE) as a loss function. We used Adam as an optimizer. We trained our epochs from 100 to 300 with a batch size of 16. The learning rate was 10^{-3}. We performed experiments on each dataset independently 6–7 times. The average result was compared with the previous state-of-the-art baselines models, as shown in Table 1. We used two real-world datasets belonging to Porto and Manhattan. We predicted the error distance rate to predict the next taxi destination with different POI at different times. The dataset is publicly available on Kaggle, and POI is extracted through the FourSquare social network.

We performed our experiments on the Porto and Manhattan datasets. We used 200,000 and 350,000 taxi rides, respectively. In both datasets, we split our model randomly for training, test, and validation sets. The splitting ratio was 70%,15%, and 20%. According to trajectory Definitions 1 and 2, the Porto and Manhattan datasets are composed of full trajectory with pick-up and drop-off points. Our dataset nature is unbalanced, and the spatial nature of the problem of calculating the stand classification means accuracy and F1-score are not appropriate. Due to this nature of the datasets, they do not give the adequate quantification of the error. We used t the Error Distance Score (EDS) defined as Haversine distance. The Haversine distance predicts the distance between two points, the current destination and actual destination of the trip.

$$EDS.kms = d_{haversine}(\hat{y}, y) \tag{15}$$

In above equation, $\hat{y}$ is the predicted location and y is the exact destination.

Table 1. The error distance score of all state-of-the-art baseline models on Porto and Manhattan datasets. The trajectory was only the pick-up and drop-off points.

Models	**Porto**	**Manhattan**
ARIMA	6.220	5.380
NN	3.215	2.375
MMLP	3.211	2.543
MMLP-SEQ	3.003	2.554
FC-LSTM	2.923	2.111
LSTM(BOC+W2V)	2.88	2.088
ST-GCN	2.67	2.00
DWSTTNs (Ours)	1.95	1.33

5.5. Hyperparameter Setting and Training

We used two real-world datasets. Wwe tuned the parameters of our model for each dataset. In Table 2, we summarize our hyperparameters settings with training and testing time. During the parameters tuning phase, we performed grid search on the number of neurons. We also adjusted the number of layers and the learning rate. The validation set was evaluated to verify the model's performance by selecting the values of the parameters, as shown in Table 2. We used the K-means algorithm for training the model with K parameters. As a fair comparison with the previous work, we set parameters 3392 for the Porto dataset and 2000 for the Manhattan dataset. It is important to note that the model is unbiased to select the number of clusters. In other words, the model selects a certain minimum level to choose the clusters. The distance between clusters and centroid is very small. As compared to the ratio of distance, the quantity of clusters is high.

After a certain limit, no further improvement occurred when adding more clusters during the training. We set the size of a layer for each feature as 10. The location of the embedding layer was equal to the size of 20. The transformer encoder and decoder worked on setting the size of FourSquare macro-categories; the size was equal to 10. The input dimension of the linear layer and scaled dot product were obtained through the vector's size. After concatenating with embedding layers and feed-forward neural networks, the vector was received, following all previous research representation to a fair comparison. We trained our model using the Adam optimizer. We trained the model with separate values of longitude and latitude. The model shares the underlying structures, such as embedding, including spatial and temporal transformers, linear layer, scaled dot product, weighted sum layers, prediction layer, dropout layer, and feed-forward layer. We trained our model with MSE as the loss function. We set checkpoints and early stopping to converge the model. The main advantage of the early stopping is that we can stop our model training if the model MSE score does not change on the validation set, such as setting a limit of 10 or 20 epochs. We computed the result of the MSE score on each epoch. During the training phase, on the validation set, we could save the network parameters if we obtained a new best MSE score. In our model architecture implementation, we used the parameters which gave the best validation MSE score during the test phase. We set the dropout rate as 0.5 to avoid the overfitting and our model with a window size of 5. We set the embedding size of 20 for dimensionality. We used Jupiter lab word embedding for the textual label using the encoder and decoder phase.

Table 2. Hyperparameter setting and training summary.

City DataSet	Batch Size	Learning Rate	Neurons	Optimizer	Activation Function	Training Time (M)	Testing Time (S)
Porto	16	10^{-3}	256	Adam	ReLU	74	o.23
Manhattan	16	10^{-3}	256	Adam	ReLU	69	0.17

5.6. Experimental Results

The performance results are compared in Table 2. We compared our results on the Porto and Manhattan datasets with the previous state-of-the-art baselines models. Our model outperforms the state-of-the-art models LSTM(BOC+W2V) and ST-GCN by a large margin. We modeled with dynamic spatial and temporal dependencies, on both of which our model consistently outperforms the previous models. Our model performance is efficient and achieves long-range temporal dependencies and hidden spatial dependencies effectively. We also observed that the performance of our model is much better than NN and simple MMLP. As shown in Table 2, NN and MMLP performances are worst for calculating the error distance score in kilometers. On the Porto dataset, our performance is 45–47% better than NN and MMLP and similar to the improved variant of MMLP-SEQ. On the Manhattan dataset, our performance is 35–40% better than NN and MMLP and the same as MMLP-SEQ. In our approach, we represent the input features constituted by the driver and time. The spatial zone embedding is the coordinates of each zone cluster fed into the embedding layer. The ARIMA model results are the worst of all baseline models.

The previous approach, FC-LSTM, and the improved variant with BOC and W2V, perform well when combined with LSTM as LSTM(BOC+W2V). The error rate of the improved version of LSTM(BOC+W2V) is lower. In the LSTM(BOC+W2V) approach, BOC and W2V indicate the recurrent neural network with features. The ST-GCN is outperformed by LSTM(BOC+W2V). The error rate more improved compared to the previous models. The transformer approach is new; being first used in 2017. The transformer encoder and decoder perform NLP translation. On the Porto dataset, we compared the performance of our approach with LSTM(BOC+W2V) and ST-GCN. Our approach performs 30–35% better than the approaches mentioned above. On the Manhattan dataset, we obtained 25–32% better results. Figure 8 illustrates the error distance score on Manhattan and Porto datasets. In Figures 9 and 10, we compare DWSTTNs in regression and multi-class classification settings, respectively. We removed two output neurons and used categorical cross-entropy as a loss function during the training of classification models. In this way, the positions of the locations were restricted to the list of centroids for clusters. However, through the corresponding probability coming from the SoftMax layer, the coordinates were weighted by each cluster centroid. The experimental results show the exact location by applying the regression procedure on the longitude and latitude values. The rate of EDS decreased on the Manhattan dataset, demonstrating the ability of the proposed method to perform excellently on cities not equally extended in longitude and latitude.

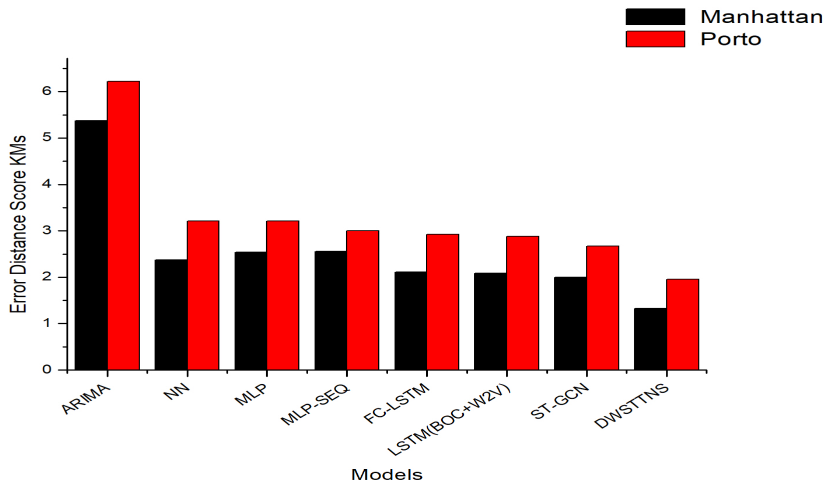

Figure 8. Porto and Manhattan error distance score comparison with previous models.

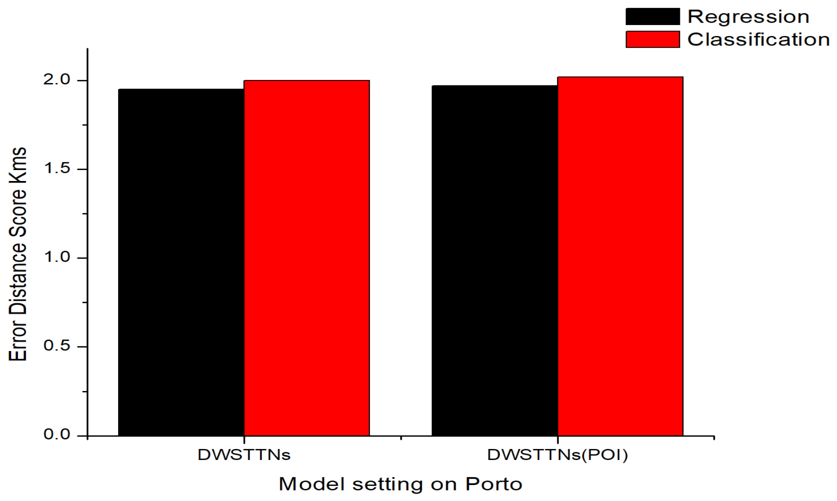

Figure 9. The Regression and classification performance on Porto dataset.

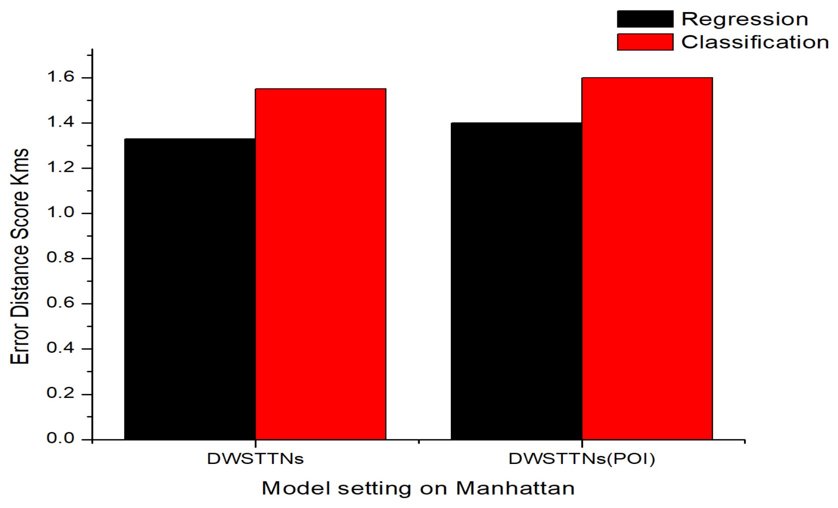

Figure 10. The Regression and classification performance on Manhattan dataset.

6. Discussion

We propose a prediction method of taxi driver next destination behavior modeling. We studied the impacts of geographical information on the accuracy of the taxi journey according to driver behavior. We obtained the following findings:

1. The problem of predicting the next destination is a well-studied application of human mobility in-real world scenarios, which finds several applications—for instance, optimizing the performance of electronic dispatching systems for predicting and reducing traffic jams. With Location-Based Social Networks (LBSN), we encode the geographical bases of visited semantic locations. In particular, we trained our model for the exact longitude and latitude coordinates to predict the next destination. We tasted our approach based on the datasets of Porto and Manhattan.
2. Based on the DWSTTNs, including spatial-temporal semantics features led to a lower MSE than the baseline method. We calculated the error distance rate on two datasets. On the Porto dataset, the error rate of the ARIMA error rate was worst, while NN and MMLP outperformed ARIMA. The two variants of LSTM, FC and BOC+W2V, performed well as compared to MMLP and NN. LSTM variants gained an improvement of 0.32 as compared to MMLP and NN. STGCN outperformed the LSTM variants; the improvement rate was decreased by 0.20. The error rate of DWSTTNs decreased by 0.72 compared to STGCN. The performance of all baseline methods is shown in Figure 8.
3. Based on the Manhattan dataset, the error rate of the ARIMA error rate was worst, while NN and MMLP outperformed ARIMA. The two variants of LSTM, FC and BOC+W2V [50], performed well compared to MMLP and NN. The LSTM variants gained an improvement of 0.42 compared to MMLP and NN. STGCN outperformed the LSTM variants; the improvement rate was decreased by 0.88. The error rate of DWSTTNs decreased by 0.67 compared to STGCN. The performance of all baseline methods is shown in Figure 8.
4. As the research on artificial intelligence and deep learning (AD) is growing, we are only beginning to see what this future might look like. From smart self-driving vehicles to AI-powered robot surgeons and smart factories, computers and machines that can learn and adapt, the world as we know it will soon change. Although we are still in the incipient phase of AI technology, billions of dollars are being spent on research and development, helping to drive AI advances. AI spending will increase by more than 50% year-on-year, reaching $57.6 billion by 2021.

6.1. Limitation of Study

This research still has some limitations. In the future, we will further consider these limitations. For example, we will increase evaluation matrices to analyze our results from different perspectives. We will merge destination problems with travel time estimation, taxi demand and supply, and origin–destination mobility on demand (MOD). In this research, our aim is to predict the next destination with the impact of geographical information, and we will increase our prediction accuracy to reduce the customer waiting time for a ride and driver waiting time to pick up a customer. Taxi companies could also optimize their management to improve their service, while urban transport planners can use this information to better plan the urban traffic. The prediction accuracy over our approach is better as compared to prior techniques, but there are still some weaknesses in real-life practice, such as road congestion and air pollution. We can use artificial intelligence and deep learning technologies to tackle these challenges.

6.2. Major Concern: The Future of Transportation and How AD Is Helping Vehicles Think

Our proposed methodology is based on destination-based AD technology [51]. We apply traffic optimization infrastructure, and, due to the efficient capabilities of AD, we collect data, analyze them, and create solutions. Our research approach is based on the next destination according to human behavior. Due to massive traffic flow and narrow human vision, traffic congestion and road accidents occur. The destination-based AD solution is developed for fully autonomous vehicles (AVs) to be used as taxis, personal transportation, commercial transportation, etc. Self-driving cars, also called AVs, are cars that operate with little to no human input. Due to the technology boom of AD, self-driving cars have been receiving incredible attention. In the past few years, AD has become the most considered research and development investment of many organizations across the world.

6.3. Cars Autonomy Levels

Self-driving has some autonomy stages referring to how much of the driving is done by the AD versus a human. We discuss the details of the stages, where higher stages illustrate more of the driving done by a computer. The stages are as follows.

- **Stage 0.** A human fully controls all functions and systems of a car.
- **Stage 1.** The computer may control minor things one at a time, for example automatic braking, cruise control, or detecting something in the blind spot.
- **Stage 2.** The car performs two simultaneous functions with the help of a computer, e.g., acceleration and steering. In this scenario, humans are still required for emergency procedures and safe operation.
- **Stage 3.** All critical operations of a car simultaneously can be controlled by the computer, including steering, accelerating, stopping, navigation, and parking under most conditions. In this scenario, in the case of any emergency, the human driver is still expected to be present.
- **Stage 4.** In some driving scenarios, the car is fully autonomous without any need for a human driver. For example, when the weather is sunny or cloudy, the car can fully drive itself but not when it is snowing and the lanes are covered.
- **Stage 5.** In every situation, the car is fully capable of self-driving. In Figure 11 illustrated all stages of car autonomous levels.

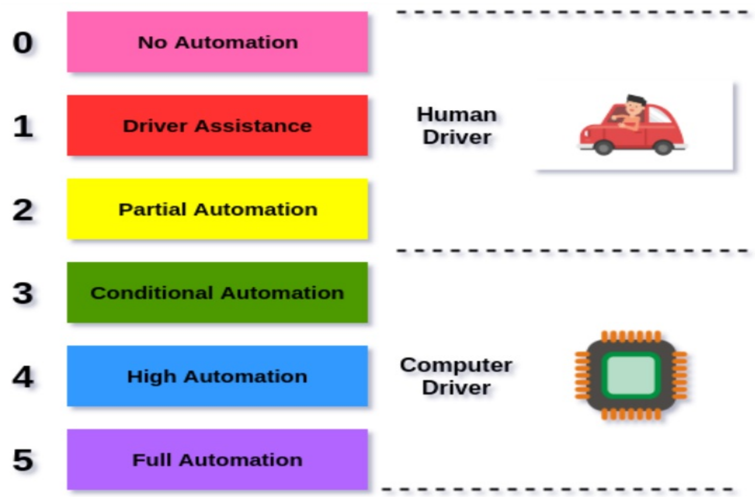

Figure 11. The stages of human driving and self driving cars.

Many researchers and engineers are working hard to use a combination of various cutting-edge software and hardware technologies to achieve self-driving. There are three use cases in a typical self-driving system: sensing, understanding, and control. More details are given in the Autonomous Vehicle Encyclopedia [52].

6.4. *Safety and Usage Optimization of Public Transportation*

- Trip planning data (public and private) is incorporated through modes and organizations, so the common public has the opportunity to assess their journey choices with detailed travel time, cost, environmental influence, etc.
- There are centrally accessible real-time agendas for all transportation approaches.
- Automobiles and transit plans are "right-sized" so that fleets are used efficiently, and no more vacant buses are available.
- Ticket payment is made remotely, and, for each entire journey, only one payment is needed.
- Generally, journey times are consistent and well-communicated.
- Many of these facilities provide benefits to those with lower incomes and people with disabilities.

6.5. *AVs Future Stand Near*

The intelligent traffic environment is no longer a science fiction proposition but in the immediate future. Buses, taxis, private vehicles, planes, trains, etc. are provided to improve the way we get around. Ford, GE, Volkswagen, Audi, Toyota, BMW, and Nissan are all at work developing and testing AVs that will be road-ready by 2020. The U.S. Secretary of Transportation stated at the 2015 Frankfurt Auto show that he expects autonomous vehicles to be in use all over the world within the next 10 years.

It is anticipated that this AD transportation revolution will make our roads and highways smoother, alleviate traffic congestion, make our transportation structures more competitive, and make transportation more enjoyable. As AVs give people more time to work and be productive, the tendency towards urbanization might be reversed.

The potential impact of AD on transportation is immense. The engineering of how we drive, produce, and ship goods on Earth and potentially in space in the future will continue to be reshaped by progress.

7. Conclusions

In this paper, we present a new way to tackle the problem of taxi driver behavior for next destination prediction with excellent performance, as compared the prior state-of-the-art framework. The previous research approaches use the whole trajectory to predict the final destination. In our approach, we rely on the individual driver rides history. The driver picks up and drops off the passenger according to their work shift; we take the last visited points of the driver. Our approach is based on the instantaneous prediction, when the driver picks up their ride and starts the ride. The resultant information is related to the dispatcher taxi company belonging to New York cities. The benefit in the real world of this kind of research is to reduce the customer waiting time for a ride, driver waiting time to pick up a customer, make the taxi demand accurate, save time, and improve fuel efficiency. The taxi companies also optimize their management to make their company have the best service. Our approach gives the accurate information according to the driver's habits. The benefit of determining the driver habit about his/her rides approach is that the urban transport planner uses this information to better plan the urban traffic. The urban transportation planner observes and monitors the taxi driver's behavioral habits and updates the traffic supply schedule and traffic roots.

We propose novel DWSTTNs, encoder, and decoder parts, demonstrating how to use the geolocated information. To the best of our knowledge, we are the first to do to present a framework that addresses this challenge as an issue of regression, rather than as a multi-class classification. We use the LBSN online social network FourSquare to embed the semantic geolocated information. The POI is embedded by the API of FourSquare and, in our approach, the API is embedded in the urban zones. Considering the urban mobility flow pattern, through pattern prediction accuracy, strongly improves the results as compared to previous approaches. In our model, we use spatial and temporal transformers. Both sub-transformers embed spatial and temporal information. The model has an encoder and a decoder. Both work with the inner layers of the transformer networks. The liner layer, scaled dot product, attention layer, and the feed-forward neural network output of the encoder are the input of the decoder. However, our study still has some limitations. We need to improve our research. In this perspective, in the future, we will work to integrate different data sources—buses, subways, etc. COVID-19 also affects the transportation system. We are working to gather datasets from different regions to visualize human mobility. We will consider more POI from FourSquare or some external sites. We will consider the POI graphs instead of isolated points to visualize human mobility and predict the next destination.

Author Contributions: Conceptualization, Z.U.A., H.S., and Z.Y.; funding acquisition, H.S.; methodology, Z.U.A., H.S., and Z.Y.; project administration, Z.U.A.; resources, H.S.; software, Z.Y. and A.A.; supervision, H.S. and Z.Y.; validation, Z.U.A., H.S., and Z.Y.; visualization, Z.U.A.; writing—original draft, Z.U.A. and A.A.; and writing—review and editing, Z.U.A., A.A., R.Z.A., and A.I. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Science Foundation of China grants 61672417 and 61876138, and Shaanxi key research and development program 2020KW-002. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

Acknowledgments: Data were retrieved from Kaggle and FourSquare.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

ITS	Intelligent Transport System
DWSTTNs	Deep wide Spatial-Temporal based Transformer Networks
LBSNs	Location Base Social Networks
GPS	Global Positioning System
SVR	Support vector regression
MLP	Multi-Layer perception
KNN	k-nearest neighbor
SVM	Support Vector Machine

SRCN	Spatial recurrent convolutionary networks
POI	Point of Interest
RNN	Recurrent Neural Network
LSTM	Long short-term memory
DMVST-Net	Deep spatial-temporal multi-view network
ARIMA	Auto-Regressive Integrated Moving Average
BiLSTM	Bidirectional Long short-term memory
CRS	Coordinate Reference1 System
NN	Nearest Neighbors
FC	Fully Connected
BOC	Bag of Concept
W2V	Word to vector
STGCN	Spatio-Temporal Graph Convolutional Networks
AD	Artificial intelligence and Deep learning
AVs	Autonomous vehicle system
SAE	Stacked autoencoder

References

1. Michele, F.; Barlacchi, G.; Pappalardo, L.; Lucchini, L.; Lepri, B. Weak nodes detection in urban transport systems: Planning for resilience in Singapore. In Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), Turin, Italy, 1–3 October 2018; pp. 472–480.
2. Besse, C,P.; Guillouet, B.; Loubes, J.; Royer, F. Destination prediction by trajectory distribution-based model. *IEEE Trans. Intell. Transp. Syst.* **2017**, *19*, 2470–2481. [CrossRef]
3. Nehal, M.; Sakr, M.A.; Mostafa, T.; El-Bahnasy, K. Review on trajectory similarity measures. In Proceedings of the 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 12–14 December 2015; pp. 613–619.
4. Yuan, X.A.; Zhang, R.; Zheng, Y.; Xie, X.; Huang, J.; Xu, Z. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE), Brisbane, Australia, 8–12 April 2013; pp. 254–265.
5. Lv, J.; Li, Q.; Sun, Q. Wang, X. T-CONV: A convolutional neural network for multi-scale taxi trajectory prediction. In Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (Bigcomp), ShangHai, China, 15–17 January 2018; pp. 82–89.
6. Di, Y.; Zhang, C.; Huang, J.; Bi, J. Serm: A recurrent model for next location prediction in semantic trajectories. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, New York, NY, USA, 6–10 November 2017; pp. 2411–2414.
7. Luis, M.-M.; Gama, J.; Ferreira, M.; Mendes-Moreira, J.; Damas, L. Predicting taxi–passenger demand using streaming data. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 1393–1402.
8. Yann, L.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444.
9. Huang, W.H.; Song, G.J.; Hong, H.K.; Xie, K.Q. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 2191–2201. [CrossRef]
10. Zhang, J.; Zheng, Y.; Qi, D.; Li, R.; Yi, X. DNN-based prediction model for spatio-temporal data. In Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Burlingame, CA, USA, 31 October–3 November 2016; pp. 1–4.
11. Xu, J.; Rahmatizadeh, R.; Bölöni, L.; Turgut, D. Real-time prediction of taxi demand using recurrent neural networks. *IEEE Trans. Intell. Transp. Syst.* **2017**, *19*, 2572–2581. [CrossRef]
12. Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; Li, Z. Deep multi-view spatial-temporal network for taxi demand prediction. *arXiv* **2018**, arXiv:1802.08714.
13. Available online: http://zhengce.beijing.gov.cn/library/192/33/50/438650/1552930/index.html (accessed on 6 December 2019).
14. Van Lint, J.W.C.; Van Hinsbergen, C.P.I.J. Short-term traffic and travel time prediction models. *Artif. Intell. Appl. Crit. Transp.* **2012**, *22*, 22–41.
15. Vlahogianni, E.I.; Karlaftis, M.G.; Golias, J.C. Short-term traffic forecasting: Where we are and where we're going. *Transp. Res. Part C Emerg. Technol.* **2014**, *43*, 3–19. [CrossRef]
16. Ke, R.; Wan, L.; Cui, Z.; Wang, Y. Two-Stream Multi-Channel Convolutional Neural Network for Multi-Lane Traffic Speed Prediction Considering Traffic Volume Impact. *Transp. Res. Rec.* **2020**, *2674*, 459–470. [CrossRef]
17. Grindey, G.J.; Massoud A.S.; Rodin, E.Y.; Garcia-Ortiz, A. Kalman filter approach to traffic modeling and prediction. In *Intelligent Transportation Systems*; International Society for Optics and Photonics: Pittsburgh, PA, USA, 1998; Volume 3207, pp. 234–240.
18. Guo, J.; Huang, W.; Williams, B.M. Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transp. Res. Part C Emerg. Technol.* **2014**, *43*, 50–64. [CrossRef]

19. Jiao, P.; Li, R.; Sun, T.; Hou, Z.; Ibrahim, A. Three revised kalman filtering models for short-term rail transit passenger flow prediction. *Math. Prob. Eng.* **2016**, *2016*, 9717582. [CrossRef]
20. Kumar, N.; Sasibhushana, A.; Rao, G.; Arasavali, N. Development of Advanced Extended Kalman Filter for Precise Estimation of GPS Receiver Position. In Proceedings of the 2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), Chennai, India, 21–23 March 2019; pp. 213–216.
21. Kalidas, A.; Ben-Akiva, M.E. Estimation and prediction of time-dependent origin-destination flows with a stochastic mapping to path flows and link flows. *Transp. Sci.* **2002**, *36*, 184–198.
22. Chen, X.; Guo, S.; Yu, L.; Hellinga, B. Short-term forecasting of transit route OD matrix with smart card data. In Proceedings of the 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, USA, 5–7 October 2011; UK Computer Measurement Group Ltd: Washington, DC, USA, 2011; pp. 1513–1518.
23. Deng, D.; Shahabi, C.; Demiryurek, U.; Zhu, L.; Yu, R.; Liu, Y. Latent space model for road networks to predict time-varying traffic. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery Special Interest Group on Management of Data: New York, NY, USA, 2016; pp. 1525–1534.
24. Karlaftis, M.G.; Vlahogianni, E.I. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transp. Res. Part C Emerg. Technol.* **2011**, *19*, 387–399. [CrossRef]
25. Ren, S.; Yang, B.; Zhang, L.; Li, Z. Traffic speed prediction with convolutional neural network adapted for non-linear spatio-temporal dynamics. In Proceedings of the 7th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, Seattle, WA, USA, 6 November 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 32–41.
26. Zhang, S.; Zhou, L.; Chen, X.; Zhang, L.; Li, L.; Li, M. Network-wide traffic speed forecasting: 3D convolutional neural network with ensemble empirical mode decomposition. *Comput.-Aided Civ. Infrastruct. Eng.* **2020**, *35*, 1132–1147. [CrossRef]
27. Lv, Y.; Duan, Y.; Kang, W.; Li, Z.; Wang, F.-Y. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transp. Syst.* **2014**, *16*, 865–873. [CrossRef]
28. Yu, H.; Wu, Z.; Wang, S.; Wang, Y.; Ma, X. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors* **2017**, *17*, 1501. [CrossRef]
29. Duan, M.; Li, K.; Yang, C.; Li, K. A hybrid deep learning CNN–ELM for age and gender classification. *Neurocomputing* **2018**, *275*, 448–461. [CrossRef]
30. Duan, Z.; Yang, Y.; Zhang, K.; Ni, Y.; Bajgain, S. Improved deep hybrid networks for urban traffic flow prediction using trajectory data. *IEEE Access* **2018**, *6*, 31820–31827. [CrossRef]
31. De Brébisson, A.; Simon, É.; Auvolat, A.; Vincent, P.; Bengio, Y. Artificial neural networks applied to taxi destination prediction. *arXiv* **2015**, arXiv:1508.00021.
32. Ma, X.; Dai, Z.; He, Z.; Ma, J.; Wang, Y.; Wang, Y. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors* **2017**, *17*, 818. [CrossRef]
33. Ma, X.; Yu, H.; Wang, Y.; Wang, Y. Large-scale transportation network congestion evolution prediction using deep learning theory. *PLoS ONE* **2015**, *10*, e0119044. [CrossRef] [PubMed]
34. Ke, J.; Zheng, H.; Yang, H.; Chen, X. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transp. Res. C Emerg. Technol.* **2017**, *85*, 591–608. [CrossRef]
35. Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; Li, Z. Deep multi-View spatial-temporal network for taxi demand prediction. In Proceedings of the 22nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 23 February 2018; pp. 2588–2595.
36. Tian, Y.; Zhang, K.; Li, J.; Lin, X.; Yang, B. LSTM-based traffic flow prediction with missing data. *Neurocomputing* **2018**, *318*, 297–305. [CrossRef]
37. Li, Y.; Yu, R.; Shahabi, C.; Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv* **2017**, arXiv:1707.01926.
38. Qian, F.; Hu, G.; Xie, J. A recurrent neural network approach to traffic matrix tracking using partial measurements. In Proceedings of the 2008 3rd IEEE Conference on Industrial Electronics and Applications, Singapore, 3–5 June 2008; pp. 1640–1643.
39. Toqué, F.; Côme, E.; El Mahrsi, M.K.; Oukhellou, L. Forecasting dynamic public transport origin-destination matrices with long-short term memory recurrent neural networks. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 1071–1076.
40. Briand, A.-S.; Etienne, C.; Martin, T.; Latifa, O. Analyzing year-to-year changes in public transport passenger behaviour using smart card data. *Transp. Res. Part C Emerg. Technol.* **2017**, *79*, 274–289. [CrossRef]
41. Nagy, A.M.; Simon, V. Survey on traffic prediction in smart cities. *Pervas. Mobile Comput.* **2018**, *50*, 148–163. [CrossRef]
42. Hlupić, T.; Oreščanin, D.; Petric, A.-M. Time series model for sales predictions in the wholesale industry. In Proceedings of the 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 28 September–2 October 2020; pp. 1263–1267.
43. Zhu, W.; Zhang, C.; Yao, S.; Gao, X.; Han, J. A spherical hidden markov model for semantics-rich human mobility modeling. *arXiv* **2020**, arXiv:2010.01986.
44. Wang, J.; Zhong, Y.; Dai, Y.; Zhang, K.; Ji, P.; Li, H. Displacement-Invariant Matching Cost Learning for Accurate Optical Flow Estimation. *arXiv* **2020**, arXiv:2010.14851. Available online: https://arxiv.org/abs/2010.14851v1 (accessed on 6 December 2020).

45. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 6 December 2017; NIPS: San Diego, CA, USA, 2017; pp. 5998–6008.
46. Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Proceedings of the 32nd Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 4 December 2019; NIPS: San Diego, CA, USA, 2019; pp. 5244–5254.
47. Bachlechner, T.; Majumder, B.P.; Mao, H.H.; Cottrell, G.W.; McAuley, J. Rezero is all you need: Fast convergence at large depth. *arXiv* **2020**, arXiv:2003.04887. Available online: https://arxiv.org/pdf/2003.04887.pdf (accessed on 6 December 2020).
48. Porto Dataset. Available online: https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i (accessed on 10 January 2015).
49. Available online: https://www.kaggle.com/c/nyc-taxi-trip-duration (accessed on 14 August 2016).
50. Rossi, A.; Barlacchi, G.; Bianchini, M.; Lepri, B. Modelling Taxi Drivers' Behaviour for the Next Destination Prediction. In Proceedings of the IEEE Transactions on Intelligent Transportation Systems, Piscataway, NJ, USA, 19 June 2019.
51. Zain, U.A.; Sun, H.; Yang, Z.; Ali, A. The Deep 3D Convolutional Multi-Branching Spatial-Temporal-Based Unit Predicting Citywide Traffic Flow. *Appl. Sci.* **2020**, *10*, 7778.
52. Wiseman, Y. Autonomous Vehicles. In *Encyclopedia of Information Science and Technology*, 5th ed.; IGI Global: Ramat Gan, Israel, 2020; Volume 1; Chapter 1; pp. 1–11 .