ATTENTION-BASED RECURRENT NEURAL NETWORK FOR URBAN VEHICLE TRAJECTORY PREDICTION

Seongjin Choi
Department of Civil and Environmental Engineering
Korea Advanced Institute of Science and Technology
benchoi93@kaist.ac.kr

Jiwon Kim
School of Civil Engineering
The University of Queensland
jiwon.kim@uq.edu.au

Hwasoo Yeo
Department of Civil and Environmental Engineering
Korea Advanced Institute of Science and Technology
hwasoo@kaist.ac.kr

December 19, 2018

ABSTRACT

As the number of various positioning sensors and location-based devices increase, a huge amount of spatial and temporal information data is collected and accumulated. These data are expressed as trajectory data by connecting the data points in chronological sequence, and thses data contain movement information of any moving object. Particularly, in this study, urban vehicle trajectory prediction is studied using trajectory data of vehicles in urban traffic network. In the previous work, Recurrent Neural Network model for urban vehicle trajectory prediction is proposed. For the further improvement of the model, in this study, we propose Attention-based Recurrent Neural Network model for urban vehicle trajectory prediction. In this proposed model, we use attention mechanism to incorporate network traffic state data into urban vehicle trajectory prediction. The model is evaluated by using the Bluetooth data collected in Brisbane, Australia, which contains the movement information of private vehicles. The performance of the model is evaluated with 5 metrics, which are BLEU-1, BLEU-2, BLEU-3, BLEU-4, and METEOR. The result shows that ARNN model have better performance compared to RNN model.

*K*eywords Vehicle Trajectory · Trajectory Prediction · Recurrent Neural Network · Attention Mechanism · Network Traffic State

1 Introduction

Recently, with abundance of various location sensors and location-aware devices, a large amount of location data are collected in urban spaces. These collected data are studied in the form of so-called moving object trajectory which is a trace of moving object in geographical spaces represented by a sequence of chronologically ordered locations [1]. Of particular interest are urban vehicle trajectory data that represent vehicle movements in urban traffic networks. Such urban vehicle trajectory data offer unprecedented opportunities to understand vehicle movement patterns in urban traffic networks by providing rich information on both aggregate flows (e.g., origin-destination matrix and cross-sectional traffic volume) and disaggregate travel behaviours including user-centric travel experiences (e.g., speed profile and travel time experienced by individual vehicles) as well as system-wide spatio-temporal mobility patterns (e.g., origin-destination pairs, routing information, and network traffic state) [2]. Previous studies have used urban vehicle trajectory data to perform travel pattern analysis [2, 3] and develop real-world applications such as trajectory-based bus arrival prediction [4] and trajectory-based route recommendation system [5].

Among many applications of trajectory data mining [6], this study focuses on trajectory-based location prediction problem. This problem concerns analyzing large amounts of trajectories of people and vehicles moving around a city to make predictions on their next locations [7, 8, 9], destinations [10, 11, 12, 13, 14], or the occurrences of traffic related events such as traffic jams and incidents [15]. In this study, we address the problem of predicting the sequence of next locations that the subject vehicle would visit, based on the information on the previous locations from the origin of the current trip and historical database representing the urban mobility patterns.

Trajectory-based location prediction is gaining increasing attention from both academia and industry because of its potential to improve the performance of many applications in multiple domains. One example is Location-based Service (LBS). LBS uses location data of service users and provide user-specific information depending on the locations of service users. Typical examples of LBS are social event recommendation, location-based advertising, and location-based incident warning system. The location prediction can be applied to predictively give information; for example, if a user's next location is expected to be disastrous or congested, the service informs the user to change route. Furthermore, when it is not possible to continue to give service because the position of the user is lost due to sensor malfunctioning, predicting the locations of the user can temporally replace the role of positioning system and continue the service [16, 17]. Another example is the application on agent-based traffic simulators. Unlike traditional traffic simulators which consider traffic demand as input, an agent-based traffic simulator requires information on individual vehicle journey such as origin, destination, and travel routes [18]. The result of vehicle location prediction can be used for real-time application of these agent-based traffic simulators. Vehicle location prediction can also be applied to inter-regional traffic demand forecasting. As the market of ride-sharing is continuously growing and Shared Autonomous Vehicles (SAV) are expected to be on our roads in the near future, there is a strong need to be able to predict inter-regional traffic demand so as to dispatch the proper number of SAV to areas of high demand. Location prediction model can be used to identify the demand hotspots by learning the mobility pattern of the users.

In our previous work [19], we proposed a Recurrent Neural Network (RNN) model to predict next locations in vehicle trajectories by adopting ideas from text generation model in natural language processing, where RNN has shown great success, and adapting them for use in our problem of location sequence prediction. The RNN model [19] considered the previously visited locations as the only input to predict the next location. Despite its simple structure, the model produced promising results. For instance, for more than 50% of all the tested trajectory samples, our RNN model showed a high prediction accuracy in that the probability of correctly predicting the next location was greater than 0.7, whereas the referenced non-RNN model (used for performance comparison) showed the similar accuracy level only for less than 5% of the tested samples [19]. To further improve the model performance, this study considers additional inputs that are likely to help predictions and proposes methodology that allows the incorporation of heterogeneous input soucres into the existing RNN framework. A specific input that we consider in this study is the surrounding traffic conditions of a vehicle at the time when it starts its journey. Nowadays, drivers can easily observe the current traffic state in the urban traffic networks and plan their journey (choose their routes) by using various traffic information and routing services [20, 21]. As a result, the location sequences (chosen routes) of individual vehicles are expected to be influenced by the network traffic conditions at the beginning of their journeys. Inspired by this idea, this study proposes an Attention-based RNN model, which embeds an attention interface to enable the RNN model to consider the current traffic state as an additional input to location prediction. A detailed explanation is in the Methodology section.

2 Methodology

2.1 Representing Urban Vehicle Trajectories as Cell Sequences

Urban vehicle trajectory refers to a sequence of locations and times describing the path that a vehicle follows along its journey in urban traffic networks. Various sensors collect the location (x, y) of vehicles and passage time(t) to form the vehicle trajectory data. The data points in vehicle trajectories are continuous in space; that is, points are continuous-scaled coordinates of longitude and latitude. To learn movement patterns from a large amount of trajectory data, however it is necessary to define a finite set of representative locations that are common to all the trajectories that have the similar path. As such, the first step in building the trajectory prediction model is to discretize the vehicle trajectory data and convert each trajectory to a sequence of discretized locations. Based on the previous studies [19, 22, 23], we partition the urban traffic network into smaller regions, or cells, so that continuous-scaled raw vehicle trajectory data are represented as discretized cell sequence data.

Let $Tr = [(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)]$ represent a raw vehicle trajectory consisting l number of data points, where data point (x_i, y_i) represents the longitude and latitude coordinate of the vehicle's i^{th} position. Using a large number of vehicle trajectory data allows the data points in all trajectories to combine and cluster in space based on the desired radius, denoted by R. Accordingly, the distance between the centroid of the point cluster and its farthest member point
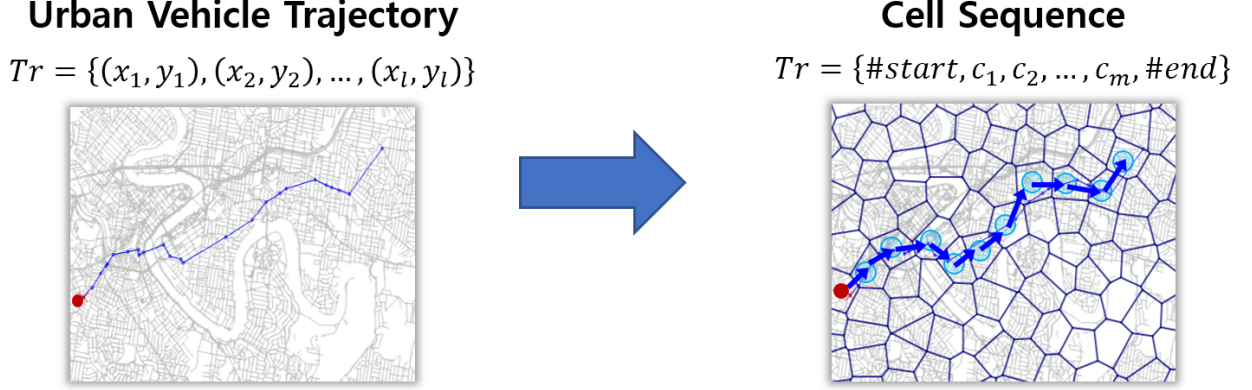
Urban Vehicle Trajectory

$$Tr = \{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$$

Cell Sequence

$$Tr = \{\#start, c_1, c_2, ..., c_m, \#end\}$$

Figure 1: Representing urban vehicle trajctory as cell sequence

is approximately R for each spatial cluster. The centroid is the mean location of the data points within the cluster, and by using Voronoi tessellation method, the cell boundaries of the clusters (Voronoi polygons) are determined.

Given N cells in the network, a vehicle trajectory can be expressed as a cell sequence $[c_1, c_2, ..., c_m]$, where c_j is the index of the jth visited cell ($1 \le c_j \le N$) within trajectory Tr. Since each of the visited cells covers multiple trajectory data points, the length of the cell sequence (m) is always less than or equal to the length of original vehicle trajectory (l) (i.e. $m \le l$). In addition to the cell sequence covering the original trajectory, two virtual spatial tokens $\#start$ and $\#end$ are added to the front and the back of the cell sequence. These virtual tokens are treated as virtual cells that do not exist in the actual network but only indicate the start and the end of the trip.

The cell sequence is then separated into input vector X and output label vector Y for training, validating, and testing. Given the cell sequence containing $m + 2$ cells including the start and the end tokens, input vector X consists of first m+1 cells and output label vector Y consists of $m + 1$ elements starting from the second element (c_1).

$$
\begin{aligned}
X &= [X_0, X_1, X_2, ..., X_m] \equiv [\#start, c_1, c_2, ..., c_m] \\
Y &= [Y_0, Y_1, Y_2, ..., Y_m] \equiv [c_1, c_2, ..., c_m, \#end]
\end{aligned}
\tag{1}
$$

2.2 Cell Sequence Prediction using Recurrent Neural Network

In our previous work [19], a Recurrent Neural Network (RNN) model for the cell sequence prediction was developed and evaluated. This previous RNN model was designed to predict the future cell sequences based purely on the previously visited cell sequence. In the training step, the model calculates the probability of each cell being visited in the next step ($\hat{Y}_i$) based on input vector X. The model structure is shown in the Fig. 2. Then, the model calculates the cross-entropy loss ($L(Y, \hat{Y})$) between the correct label (Y) and the predicted label probability based on the current parameter ($\hat{Y}$). A basic Long Short Term Memory (LSTM) unit [24] is used as the hidden unit in the RNN model, i.e., RNN units in Fig. 2. For parameter estimation, the Adam optimizer was used to update the model parameters [25]. A detailed explanation of the earlier model can be found in our previous paper [19].

2.3 Incorporating Network Traffic State Data into Cell Sequence Prediction

The drivers can easily obtain the current traffic state in the urban traffic networks, and plan their journey by using various traffic information and routing services [20, 21]. For example, between routes A and B, a driver is likely to choose route A when route B is congested and vice versa. As a result, the location sequences (chosen routes) of individual vehicles are expected to be influenced by the network traffic conditions at the beginning of their journeys. It is thus desriable to incorporate network-wide traffic state information and route choice behavior depending on the prevailing traffic state into the RNN-based cell sequence prediction model to increase the model's prediction accuracy.

Adding additional information to RNN models, which is network traffic state in our case, is not a straightforward task. RNN models are specialied to process sequential data considering temporal dependency across time or sequence steps. When input data are all in the form of sequence, adding another sequence input can be done through a straightforward extention as RNN model can have multiple input layers and multiple hidden features to incorporate multiple sequence
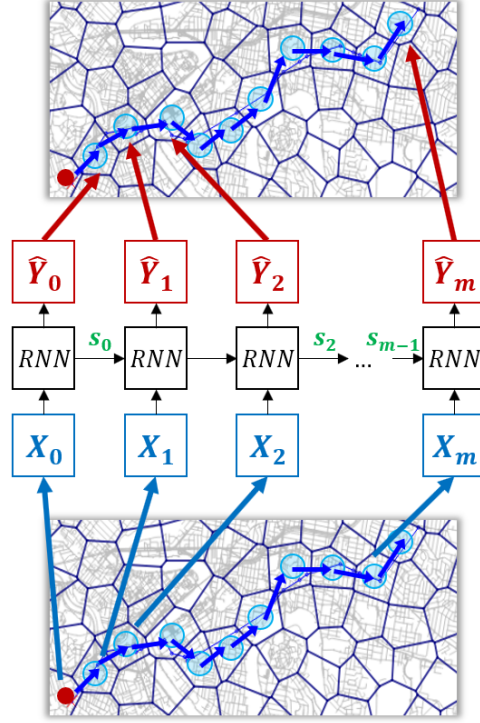
Figure 2: Structure of the basic Recurrent Neural Network model (RNN) for cell sequence prediction

inputs and combine them to calculate the output. However, when the additonal input is non-sequential data, it cannot be directly represented as an input layer of the RNN model but rather should be processed outside the RNN model. The traffic state information we wish to add as an additional input to our RNN model is network-wide traffic density level at the beginning of the sequence, i.e., traffic information available at the origin (X_0) of a given trajectory), which is non-sequential data, making the problem more challenging. It may be possible to generate network traffic state data in a sequential form by feeding network traffic state at the time that a subject vehicle visits each cell in cell sequence. However, it requires a model that preidicts the location and visiting time simultaneously and that is beyond the scope of our current study, which focuses on location prediction only.

One way to address this challenge is to introduce attention mechanism. The attention mechanism can be understood as an interface between external information processed outside the RNN model and sequential inputs processed inside the RNN model, as illustrated in Fig. 3. The attention mechanism in neural networks was first introduced to imitate the "attention mechanism" in human brain. When humans are asked to translate a sentence from one language to other language, humans try to think of words that match the alignment and meaning of word while also considering the global context of the sentence. Similarly, when humans are asked to write a sentence based on an image, humans not only concentrate on the important part of the image but also think of the global context of the image to write a sentence. Using the attention mechanism in neural networks has shown significant improvements in model performance in applications such as machine language translation [26] and video captioning [27].

The attention mechanism allows the cell sequence prediction model or cell sequence generator to concentrate on certain part of the network traffic state input and use the information for cell sequence generation. There are mainly two tasks given to the attention mechanism: first is to set initial state for the RNN and second is to provide the network-wied traffic state information at each cell generation step. Usually, the initial state vector of RNN cell is set as zero vector since the simplest form of RNN does not consider additional information from other models or inputs. However, in the case of ARNN, there is an additional information of network traffic state. To consider this input in cell sequence generation, this information should be embedded into the model. Also, the attention mechanism allows the RNN to consider the traffic state in predicting the next location, or cell, at each step. The model is trained to calculate which information, or which region, to consider among the network traffic state data by calculating the context vector and attention weights.
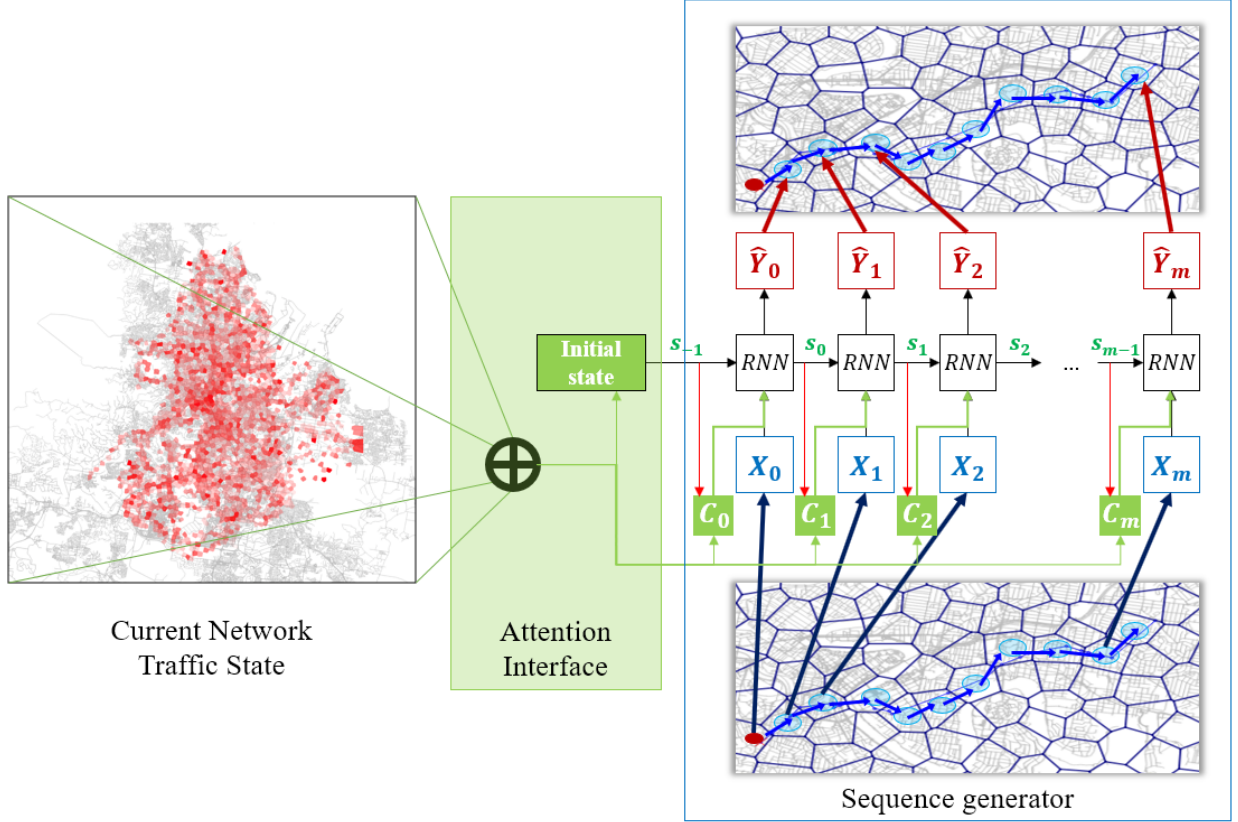
Figure 3: Structure of the proposed Attention-based Recurrent Neural Network model (ARNN) for cell sequence prediction

Fig. 3 shows the structure of the Attention-based Recurrent Neural Network (ARNN) model for the cell sequence prediction. There are two types of input data in this model: the first is the network traffic state data and the second is the cell sequence representation of vehicle trajectory data. The model first processes the current network traffic state and calculates the initial state (s_{-1}) for the RNN unit. Then, the attention interface calculates the context vector (C_i) based on the previous state vector. The context vector (C_i) is used as input to the ith RNN unit as well as the corresponding input vector element (X_i) to update current state vector (s_i). The attention weight $\alpha_{(i,j)}$ is calculated based on the context vector and previous state vector ($\alpha_{(i,j)} = f(C_i, s_{(i-1)})$). The attention weight represents the probability to attend to j^{th} cell at i^{th} sequence. Therefore, the sum of $\alpha_{(i,j)}$ at each sequence is 1 ($\sum_{\forall j} \alpha_{(i,j)} = 1$).

The input cell sequence (X) is processed based on the word-embedding method to represent the hidden features of the cells. In the training step, input vector X is directly used as an input of each RNN unit in order to calculate the output vector ($\hat{Y}_i$). However, in the testing step, only the front n cell sequence elements are directly used. Afterward, since the output vector represents the probability of each cell being visited, we use a random sampling based on the multinomial distribution with probability $\hat{Y}_i$ to extract the next cell, also it is used as the next input vector element.

A basic Long Short Term Memory (LSTM) cell [24] is used as RNN cell. And the model also uses the Adam optimizer to update the model parameters [25].

3 Model Performance Evaluation

3.1 Data

3.1.1 Urban Vehicle Trajectory Data

The vehicle trajectory data used in this research are collected from the Bluetooth sensors in Brisbane, Australia, provided by Queensland Department of Transport and Main Roads (TMR) and Brisbane City Council (BCC). The

(a) Vehicle accumulation
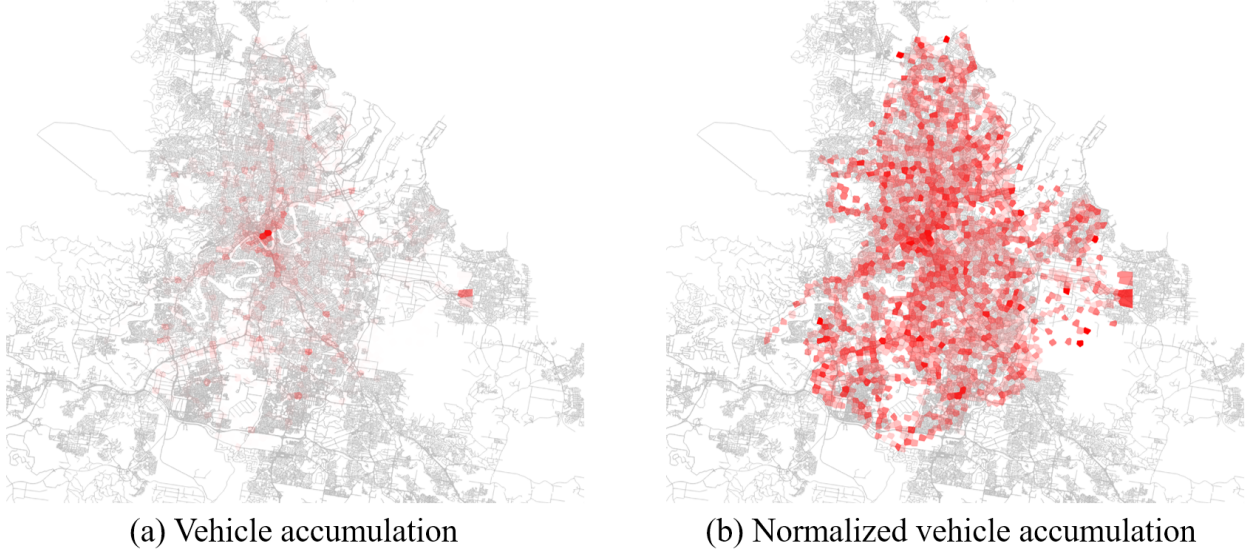
(b) Normalized vehicle accumulation

Figure 4: Spatial distribution of (a) vehicle accumulation, and (b) normalized vehicle accumulation at 12pm on March 1, 2016.

Bluetooth sensors are installed in state-controlled roads and intersections located inside the Brisbane City, and they detect Bluetooth devices (e.g., in-vehicle navigation systems and mobile devices) passing the sensors and record their passage time. By connecting the data points containing the same identifier of the Bluetooth device (MAC ID), the vehicle trajectories of individual vehicles can be constructed. Each vehicle trajectory represents a time-ordered sequence of Bluetooth sensor locations that a subject vehicle passes. If the corresponding vehicle does not move for more than an hour, it is considered that the vehicle trip has terminated. For this case study, we used the vehicle trajectory data collected in March 2016. There are approximately 276,000 trajectories in one day, and a total of 8,556,767 vehicle trajectories are collected in March 2016. We randomly sampled 200,000 vehicle trajectories for the training dataset, 10,000 vehicle trajectories for the validation dataset (used in hyper-parameter searching), and 200,000 vehicle trajectories for the testing dataset.

Brisbane urban traffic network is divided into "cells" to use the vehicle trajectory clustering and cell partitioning method proposed in the previous research [22, 23]. The desired radius of the cells is set to be 300m. Accordingly, a total of 5,712 cells are generated. Among them, 2,746 cells are considered to be active since the rest of the cells are not visited by any vehicles in the historical data of vehicle trajectories. The vehicle trajectory data are processed and transformed into cell sequence data.

3.1.2 Network Traffic State Data

There are several ways to represent the network traffic state such as density and average speed. In this study, vehicle accumulation, which is understood as the density of cells, is used to represent the network traffic state. The vehicle accumulation for a given cell is estimated by counting the number of vehicles that are present within the cell at a given instant point in time. We processed the vehicle trajectory data and calculated the vehicle accumulation of each cell at each minute. The vehicle accumulation data are normalized by dividing the vehicle accumulation by the historical maximum number of vehicle accumulation in each cell.

The vehicle accumulation data are used as the network traffic state input to the ARNN model. When the ARNN model is trainend through each cell sequence, the model receives the vehicle accumulation data on a whole network from 10 minutes before the start time of the sequence. As a result, the shape of the input vehicle accumulation data is [N,10], where N is the number of cells in the study network.

3.2 Hyperparameter and Model Training

For each model, we applied hyperparameter searching algorithm to ensure that each model is trained to acheive its maximum performance. The hyperparameter searching algorithm used in this study is from a Python package called "Scikit-optimize [28]." The hyperparameter searching algorithm is based on Bayesian optimization using Gaussian Process (GP). This algorithm approximates the function by assuming that the function values follow a multivariate

Gaussian. The covariance of the function values is given by a GP kernel between the parameters. Then a smart choice to choose the next parameter to evaluate can be made by the acquisition function over the Gaussian prior which is much quicker to evaluate.

There are three hyperparameters to search: learning rate, embedding layer dimension, and hidden layer dimension. The learning rate determines the updating step-size at each training step. If it is too large, the model is unlikely to converge. On the other hand, if it is too small, the speed of convergence is too slow, and the model is likely to fall into local minima. Therefore, finidng an appropriate learning rate is crucial in learning neural networks. The embedding layer dimension is used to convert the cell sequence input which is treated as one-hot vector to a vector in the latent space. In other words, the embedding layer extracts feature of each cell input and represents it as a numeric vector. The hidden layer dimension determines the dimension of LSTM cells and cell decoding layer. LSTM cell is used to calculate the state vector (s_i) and cell decoding layer is used to calculate the cell-visiting probability ($\hat{Y}_i$) from the state vector (s_i).

The models are trained for 10 epochs for each hyperparameter set and the prediction accuracy is measured by applying the trained models to the validation dataset. The result of hyperparameters are shown in Table 1.

Table 1: Hyperparameter result

Model	Learning rate	Dimension of Embedding Layer	Dimension of Hidden Layer
RNN	6.216234e-05	413	854
ARNN	5.842804e-04	659	574

4 Result

4.1 Score based Evaluation of Generated Cell Sequences

In this study, two widely used evaluation metrics in sequence modeling, BLEU score and METEOR score, are used to evaluate the performance of the models (RNN and ARNN). In the previous study [19], we used the complementary cumulative distribution function of the probability to measure how accurately the model predicts the next 1, 2, or 3 consecutive cells. While this measure is intuitive and easy to interpret, there is a drawback in this method in that it considers element-wide prediction accuracy and does not take into account the whole sequence. The element-wide performance measure can be sensitive to small local mis-predictions and tend to underestimate the performance of the model. For example, when the original cell sequence is $[\#start, c_1, c_2, c_3, \#end]$ and a model is asked to predict the next cells based on the given cell sequence $[\#start, c_1]$, the prediction of $[c_2, c_4, c_3, \#end]$ will be considered as incorrect and performing poorly by our previous method because the model miss-predicted one cell c_4, even though the overall sequence is very similar to the original sequence. As such, this study employs BLEU score and METEOR score that consider the whole sequence and thus more robust and accurate as performance measure for sequence modeling.

4.1.1 BLEU score

$BLEU$ score [29] is one of the most widely used evaluation score in sequence modeling, especially in natural language processing. $BLEU$ uses modified form of precision to compare a reference sequence and a candidate sequence. For each cell in the candidate (generated) cell sequence, the algorithm takes the maximum count (m_max) of the cell in the reference cell sequence.

$$P = \frac{\min(m, m_{max})}{w_t} \tag{2}$$

where m refers to the number of a single cell in candidate cell sequence which are also found in the reference cell sequence, m_max is sum of maximum number of each cell in reference sequence which are in candidate sequence, and w_t refers to the number of cells in candidate sequence.

Basically, the $BLEU$ score uses a single cell as a unit for score calculation. If the sequences are divided by multiple cells (n cells), we calculate the $BLEU$ score for each n-cell-unit. This measures whether the model can predict the next cells consecutively. The $BLEU - n$ score is the average of $BLEU$ scores, in which the sequence is divided at all orders from 1 to n.

4.1.2 METEOR score

$METEOR$ [30] first creates an alignment between candidate cell sequence and reference cell sequence. The alignment is a set of mappings between the most similar cells. Every cell in the candidate sequence should be mapped to zero or one cell in the reference sequence. $METEOR$ chooses an alignment with the most mappings and the fewest crosses (fewer intersection between mappings).

To calculate $METEOR$ score, we first define precision P and recall R.

$$P = \frac{\min(m, m_{max})}{w_t} \tag{3}$$

$$R = \frac{\min(m, m_{max})}{w_r} \tag{4}$$

where m refers to the number of single cells in candidate cell sequence which are also found in the reference cell sequence, m_{max} refers to the sum of maximum number of each cell in reference cell sequence which are in candidate cell sequence, w_t refers to the number of cells in candidate cell sequence, and w_r refers to the number of cells in reference cell sequence.

Then, we calculate the weighted harmonic mean between precision and recall, where the ratio of the weights is 1:9.

$$F_{mean} = \frac{10}{\frac{1}{P} + \frac{9}{R}} = \frac{10PR}{R + 9P} \tag{5}$$

To account for congruity with respect to a longer cell segment that appears both in reference and candidate cell sequences, we generate mappings based on the longer cell segment and use it to compute the penalty p. The more mappings there are, which are not adjacent in the reference and the candidate cell sequence, the higher the penalty will be. The penalty is calculated as follows:

$$p = 0.5(\frac{c}{u_m})^3 \tag{6}$$

where c is a set of single cells that are not adjacent in the candidate and refernce sequence, and u_m is the number of single cells that have been mapped. This penalty reduces F_{mean} up to 50% and calculate the $METEOR$ score (M).

$$M = F_{mean}(1 - p) \tag{7}$$

4.2 Score Result

For each sequence in the test dataset, the scores are calculated by the following procedure.

Let Tr be the subject cell sequence with length m, which is expressed as:

$$Tr = [\#start, c_1, c_2, ..., c_m, \#end] \tag{8}$$

The subject cell sequence is divided into 2 parts: The sequence given $((Tr)_{1:g})$ and the sequence to be predicted $((Tr)_{(g+1:m)})$, where g is the number of cells given to the models (ARNN and RNN).

$$Tr_{1:g} = [\#start, c_1, c_2, ..., c_g] \quad Tr_{g+1:m} = [c_{(g+1)}, ..., c_m, \#end] \tag{9}$$

Each model predicts 100 candidate cell sequences based on $Tr_{1:g}$ producing a set of 100 $(Tr)_{g+1:m}$ sequences for each $(Tr)_{1:g}$. The generated candidate cell sequences are the cell sequences that have #end token at the end, representing that the trip has terminated. These candidate cell sequences may not have the same length with the original cell sequence. The length can be longer or shorter depending on when the model predicts #end token. The $Tr_{g+1:m}$ is used as reference cell sequence to calculate the score presented above. For each score ($BLEU - 1$, $BLEU - 2$, $BLEU - 3$, $BLEU - 4$, and $METEOR$), 100 score values are calculated based on the generated 100 candidate cell sequences. The average value of each score is used to represent the model performance of the corresponding cell sequence (Tr).

10,000 cell sequences in test dataset is used to calculate the five scores, which are $BLEU - 1$, $BLEU - 2$, $BLEU - 3$, $BLEU - 4$, and $METEOR$. Fig. 5 shows the score result of each model. The x-axis represents the original length of the sequence, and the y-axis represents the value for each score metrics. The result of ARNN model (red color) shows better performance compared to the result of RNN model (blue color).

The performance gap between two models tends to decrease as the original length of the sequence increases. Fig. 6 shows the result in terms of score improvement rate. The score improvement rate is defined as ratio of the performance score of ARNN model to the performance score of RNN model $(= (score_{ARNN})/(score_{RNN}))$. For each number of given cell sequence (g) and original length of cell sequence (m), this performance improvement rate is measured. And the Fig. 6 shows the summarized result. Points in Fig. 6 represents the average performance improvement rate for each original length of cell sequence (m), and the line represents the range of this value (from the minimum value to the maximum value).

The result shows that ARNN model can predict short cell sequences more accurately up to 12%. ARNN model outperforms the RNN model in terms of both $BLEU$ score and $METEOR$ score. It is worth noting that the ARNN model had performance improvement in terms of $METEOR$ score as a high $METEOR$ score not only requires good prediction of visited cells but also accurate description of cell alignment (the visiting order of cells). The result thus confirm that the ARNN model using the attention mechanism achieves improvements in predicting the composition of cells in the sequence accurately as well as the alignment of the cells in the sequence.

In Fig. 6, one interesting observation is that the performance improvement rate decreases and converges to 1 (the black lines) as the original length of the cell sequence increases. This can be because the input feature given to the ARNN model is the network traffic state at the beginning of each trajectory journey. This observation has an important implication for the influence of pre-trip information in route choice behaviors. The fact that the ARNN improves the prediction at the early stages of a journey implies that the pre-trip information indeed influences travellers' route choice decisions and differentiates route choice patterns between different pre-trip traffic conditions. The fact that the effect of pre-trip information fades away at the later stages of the journey may be the indication of drivers' reliance on en-route trip information instead of pre-trip information and thus indicates a need for incorporating such en-route information into the model to further improve the model performance.
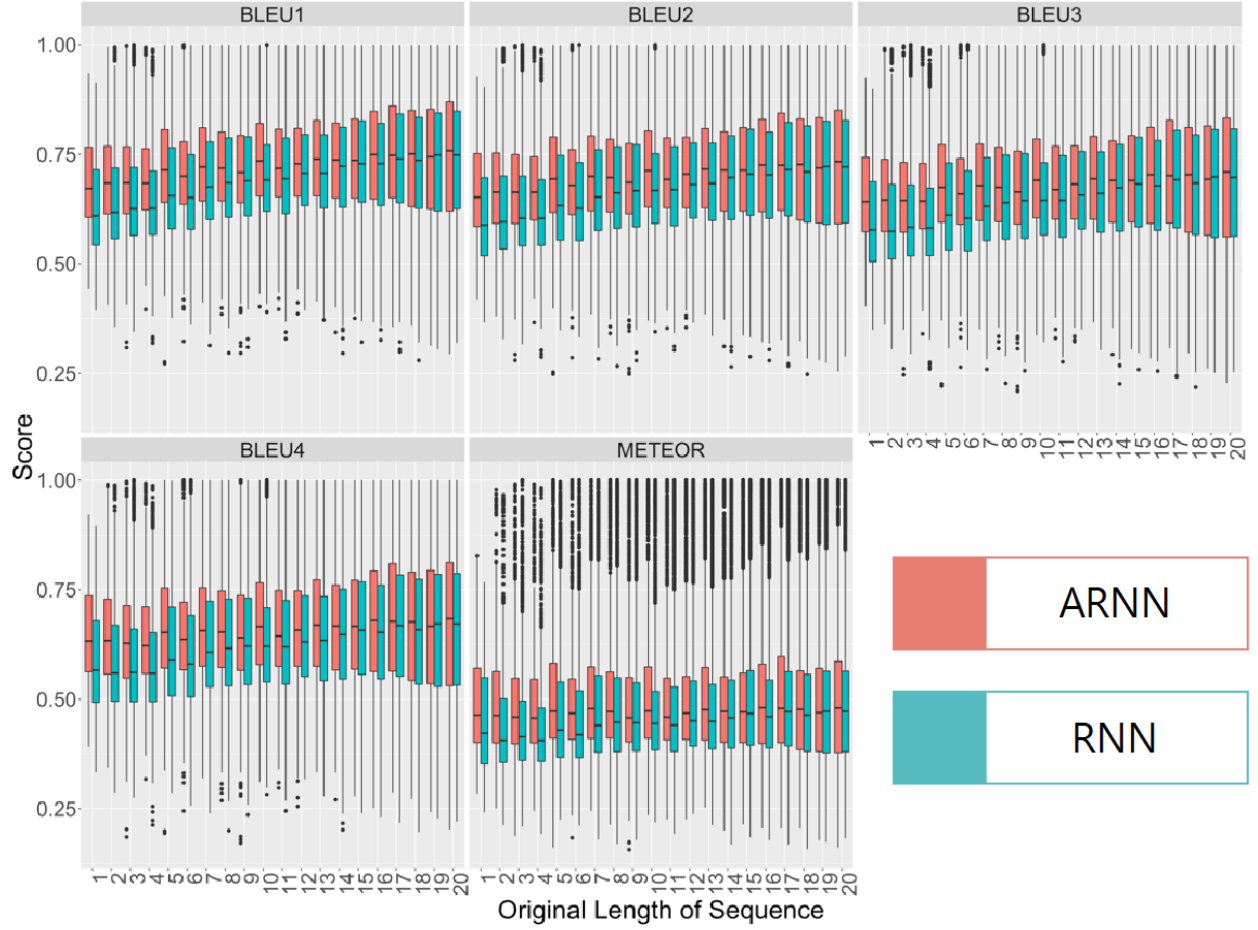
Figure 5: Boxplot of models (ARNN, RNN) for each original length of sequence (m)
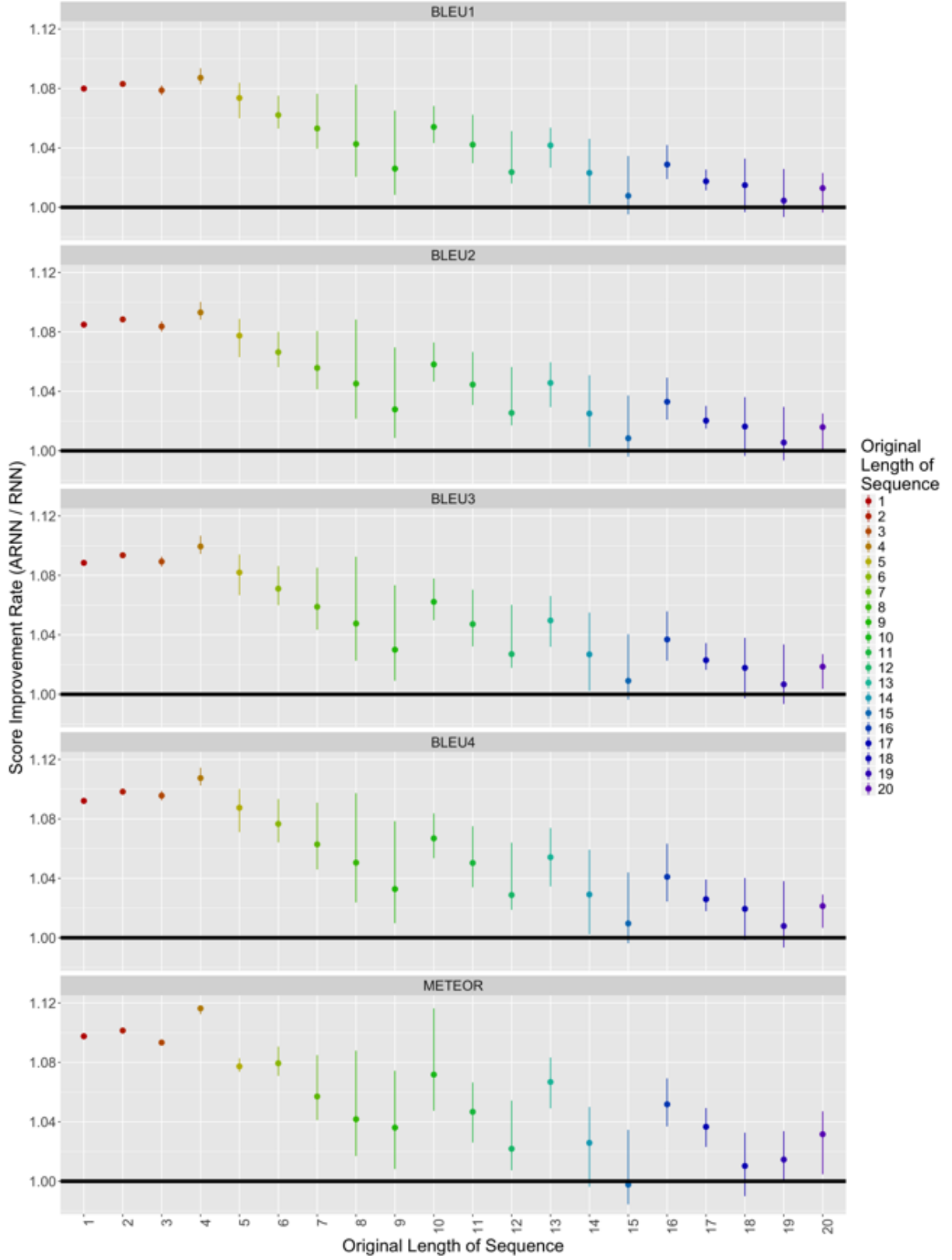
Figure 6: Score improvement rate for each original length of sequence (m). The points represent the average value, and the lines represent the range of the score improvement rate (from the minimum value to the maximum value)

5   Conclusion and Future Studies

This research studies urban vehicle trajectory prediction, one of the applications of trajectory data mining. Based on the previous work [19], in this study, we proposed a novel approach to incorporate network traffic state data into urban vehicle trajectory prediction model. Attention mechanism is used as an interface to connect the network traffic state input data to the vehicle trajectory predictor proposed in the previous work. ARNN model, which is Attention-based RNN model for cell sequence prediction, is compared with RNN model, which is RNN model for cell sequence prediction, in terms of conventional scoring methods in sequence prediction. The result shows that ARNN model outperformed RNN model. The result shows that it is effective to use attention mechanism to structurally connect the network traffic state input to RNN model to predict the vehicle's future locations. Especially, it is promising that the ARNN model showed significant performance improvement in terms of METEOR which considers not only the cells to be visited but also the alignment of the cells in sequence. The performance improvement rates tend to decrease and converge to 1 as the original number of cell sequence increase. For the further improvement of the ARNN model, this problem should be studied to maintain the performance improvement rate at steady level.

There are some limitations in this study, so further works should cover such topics. First of all, in this study, the network traffic state data were normalized by using the historical maximum value of each cell. This makes easy to represent the network traffic state, but this may lead to some problems that normalized data of the cells with very low traffic may be too sensitive to small number of vehicles and count it as heavy congestion. This makes the model overreact to these cells and makes the cell sequence prediction confused. For the further improvement of this study, different types of normalization methods should be tested. Second, although the performance measures ($BLEU$ and $METEOR$) are widely used in the fields studying sequence prediction such as natural language processing, the application of these metrics is new in the transportation domain. The interpretation and implication of these metrics in the context of traffic modeling should be further investigated and proposed.

References

[1] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[2] Jiwon Kim and Hani S. Mahmassani. Spatial and temporal characterization of travel patterns in a traffic network using vehicle trajectories. *Transportation Research Procedia*, 9:164–184, 2015.

[3] Mehmet Yildirimoglu and Jiwon Kim. Identification of communities in urban mobility networks using multi-layer graphs of network traffic. *Transportation Research Procedia*, 27:1034–1041, 2017.

[4] John Zimmerman, Anthony Tomasic, Charles Garrod, Daisy Yoo, Chaya Hiruncharoenvate, Rafae Aziz, Nikhil Ravi Thiruvengadam, Yun Huang, and Aaron Steinfeld. Field trial of tiramisu: crowd-sourcing bus arrival times to spur co-design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1677–1686. ACM, 2011.

[5] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence. *IEEE Trans. Knowl. Data Eng.*, 25(1):220–232, 2013.

[6] Jean Damascène Mazimpaka and Sabine Timpf. Trajectory data mining: A review of methods and applications. *Journal of Spatial Information Science*, 2016(13):61–99, 2016.

[7] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. Mining user mobility features for next place prediction in location-based services. In *Data mining (ICDM), 2012 IEEE 12th international conference on*, pages 1038–1043. IEEE, 2012.

[8] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, page 3. ACM, 2012.

[9] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden Markov models. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, pages 911–918. ACM, 2012.

[10] John Krumm and Eric Horvitz. Predestination: Inferring destinations from partial trajectories. In *International Conference on Ubiquitous Computing*, pages 243–260. Springer, 2006.

[11] John Krumm and Eric Horvitz. Predestination: Where do you want to go today? *Computer*, 40(4), 2007.

[12] Eric Horvitz and John Krumm. Some help on the way: Opportunistic routing under uncertainty. In *Proceedings of the 2012 ACM conference on Ubiquitous Computing*, pages 371–380. ACM, 2012.

[13] Andy Yuan Xue, Jianzhong Qi, Xing Xie, Rui Zhang, Jin Huang, and Yuan Li. Solving the data sparsity problem in destination prediction. *The VLDB Journal—The International Journal on Very Large Data Bases*, 24(2):219–243, 2015.

[14] Brian D. Ziebart, Andrew L. Maas, Anind K. Dey, and J. Andrew Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 322–331. ACM, 2008.

[15] Guangxing Wang and Jiwon Kim. The Prediction of Traffic Congestion and Incident on Urban Road Networks Using Naive Bayes Classifier. In *Australasian Transport Research Forum (ATRF), 38th, 2016, Melbourne, Victoria, Australia*, 2016.

[16] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–646. ACM, 2009.

[17] Miko\laj Morzy. Mining frequent trajectories of moving objects for location prediction. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 667–680. Springer, 2007.

[18] Luis M. Martinez, Gonçalo HA Correia, and José M. Viegas. An agent-based simulation model to assess the impacts of introducing a shared-taxi system: an application to Lisbon (Portugal). *Journal of Advanced Transportation*, 49(3):475–495, 2015.

[19] Seongjin Choi, Hwasoo Yeo, and Jiwon Kim. Network-Wide Vehicle Trajectory Prediction in Urban Traffic Networks using Deep Learning. *Transportation Research Record*, page 0361198118794735, 2018.

[20] J. L. Adler. Investigating the learning effects of route guidance and traffic advisories on route choice behavior. *Transportation Research Part C: Emerging Technologies*, 9(1):1–14, 2001.

[21] Théophile Cabannes, Marco Antonio Sangiovanni Vincentelli, Alexander Sundt, Hippolyte Signargout, Emily Porter, Vincent Fighiera, Juliette Ugirumurera, and Alexandre M. Bayen. The impact of GPS-enabled shortest path routing on mobility: a game theoretic approach 2. Technical report, 2018.

[22] Jiwon Kim, Kai Zheng, Sanghyung Ahn, Marty Papamanolis, and Pingfu Chao. Graph-based analysis of city-wide traffic dynamics using time-evolving graphs of trajectory data. In *Australasian Transport Research Forum (ATRF), 38th*, 2016.

[23] Jiwon Kim, Kai Zheng, Jonathan Corcoran, Sanghyung Ahn, and Marty Papamanolis. Trajectory Flow Map: Graph-based Approach to Analysing Temporal Evolution of Aggregated Traffic Flows in Large-scale Urban Networks. Technical report, 2017.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, \Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[27] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[28] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, fcharras, Zé Vinícius, cmmalone, Christopher Schröder, nel215, Nuno Campos, Todd Young, Stefano Cereda, Thomas Fan, Justus Schwabedal, Hvass-Labs, Mikhail Pak, SoManyUsernamesTaken, Fred Callaway, Loïc Estève, Lilian Besson, Peter M. Landwehr, Pavel Komarov, Mehdi Cherti, Kejia (KJ) Shi, Karlson Pfannschmidt, Fabian Linzberger, Christophe Cauet, Anna Gut, Andreas Mueller, and Alexander Fabisch. scikit-optimize/scikit-optimize: v0.5.1 - re-release, February 2018.

[29] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[30] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.