electronics

MDPI

Article

Context-Aware Link Embedding with Reachability and Flow Centrality Analysis for Accurate Speed Prediction for Large-Scale Traffic Networks

Chanjae Lee [1] and Young Yoon [2,3,*]

[1] Department of Artificial Intelligence, Big Data, Hongik University, 94 Wausan-ro, Sangsu-dong, Mapo-gu, Seoul 04068, Korea; arisel117@gmail.com
[2] Department of Computer Engineering, Hongik University, 94 Wausan-ro, Sangsu-dong, Mapo-gu, Seoul 04068, Korea
[3] Neouly Incorporated, 94 Wausan-ro, Sangsu-dong, Mapo-gu, Seoul 04068, Korea
* Correspondence: young.yoon@hongik.ac.kr; Tel.:+82-2-320-1659

Received: 26 September 2020; Accepted: 28 October 2020; Published: 29 October 2020

Abstract: This paper presents a novel method for predicting the traffic speed of the links on large-scale traffic networks. We first analyze how traffic flows in and out of every link through the lowest cost reachable paths. We aggregate the traffic flow conditions of the links on every hop of the inbound and outbound reachable paths to represent the traffic flow dynamics. We compute a new measure called traffic flow centrality (i.e., the Z value) for every link to capture the inherently complex mechanism of the traffic links influencing each other in terms of traffic speed. We combine the features regarding the traffic flow centrality with the external conditions around the links, such as climate and time of day information. We model how these features change over time with recurrent neural networks and infer traffic speed at the subsequent time windows. Our feature representation of the traffic flow for every link remains invariant even when the traffic network changes. Furthermore, we can handle traffic networks with thousands of links. The experiments with the traffic networks in the Seoul metropolitan area in South Korea reveal that our unique ways of embedding the comprehensive spatio-temporal features of links outperform existing solutions.

Keywords: link embedding; traffic speed prediction; traffic flow centrality; reachability analysis; spatio-temporal data; artificial neural network; deep learning; context-awareness

1. Introduction

Accurate prediction of speed on traffic networks helps improve traffic management strategies and generate efficient routing plans. However, precisely estimating the traffic speed in advance has been a non-trivial task since various factors determine traffic flows in many ways.

Various approaches have been used for traffic speed prediction using statistical methods [1–7] and machine learning with neural networks with deep hidden layers [8–18]. However, the existing solutions are mostly limited to estimating traffic speed for either a single road link or a small-scale sub-network with only a handful of traffic links (e.g., a crossroad). In reality, a road system is a large-scale connected graph with the traffic links affecting each other's traffic flow over time in a more complicated fashion that cannot be explained easily with a simple statistical model. The works without a broader view of the traffic links may not adequately unravel the hidden, but critical speed prediction factors.

In this paper, we adapt the node embedding techniques introduced by Hamilton et al. [19]. We represent the relationship between links with a feature vector whose size is invariant even when certain changes are made to the traffic networks' structure, such as the addition or deletion of roads.

We analyze how the traffic flows in and out of a link through reachable multi-hop paths computed with the Floyd–Warshall algorithm [20]. How the links impact each other given the traffic flow analysis is quantified through a novel metric we refer to as Traffic Flow Centrality (TFC). We combine the data related to TFC with the external conditions around each link, such as climate and time information (e.g., time of day, the indication of holidays). We use a recurrent neural network algorithm to correlate between the composite feature's temporal transition and the traffic speed of each link. Our method captures the traffic flow dynamics through sub-networks around every link without embedding the entire adjacency matrix. Therefore, our method is much more space efficient, and it can handle large-scale traffic networks with thousands of links. We also do not face the problem of considering irrelevant information such as the hollow points around traffic networks when they are represented either with a sparse adjacency matrix or raster graphics [10,18]. Our solution assesses the impact of the remote links in addition to the adjacent neighbors on traffic flows. Hence, with the broader contextual view and the exclusion of unnecessary information, we expect to outperform the works that based their speed prediction only on the limited view of adjacent links.

This paper is structured as follows: In Section 2, we first review the related work. In Section 3, we introduce the method for inferring traffic speed given the temporal transitions of links' composite contextual features that are modeled with a novel link embedding technique. In Section 4, we benchmark the performance of our approach against existing works, and we conclude in Section 5.

2. Related Works

In this section, we put our work in the context of various related research works on traffic speed prediction. Recently, artificial neural networks with deep hidden layers have gained popularity, as they are effective at modeling the non-linear traffic speed dynamics. Some of the notable works have employed FNNs (Fuzzy Neural Networks) [8,9], DNNs (Deep Neural Networks) [11,21], RNNs (Recurrent Neural Networks) [13,14], DBNs (Deep Belief Networks) [12,15], and the IBCM-DL (Improved Bayesian Combination Model with Deep Learning) [17] models. These works have shown more accurate speed prediction than the approaches that are based on classic statistical methods such as ARIMA (Auto-Regressive Integrated Moving Average) models [1–3], SVR (Support Vector Regression) [4,5], and K-NN (K-Nearest Neighbor) [6,7].

When the models are obtained by learning the pattern on a single specific link [1–3,13–15,17], the distinct features of other links may not be adequately accounted for. Thus, modeling the traffic speed pattern per individual link was discussed in [22]. Nonetheless, the work by Kim et al. [22] still did not reflect the substructure of the traffic network around each link.

The works presented in [10,16,23,24] extracted spatial features from a visual representation of traffic networks. In particular, Zheng et al. [23] used a two-dimensional traffic flow that is embedded in Convolutional Neural Networks (CNN). They also used a Long Short-term Memory (LSTM) [25] algorithm to model long-term historical data. Similarly, Du et al. [16] represented the passenger flows among different traffic lines in a transportation network into multi-channel matrices with deep irregular convolutional networks. Guo et al. [10] analyzed the congestion propagation patterns based on the traffic observations recorded at fixed intervals in time and fixed locations in space. They represented the traffic observation in raster data. Then, all the raster data were fed into a 3D convolutional neural network to model the spatial information. They used a 3×3 convolutional kernel that includes the hollow points where no road lies and no traffic flows. The hollow points in the convolutional kernel may accidentally reflect stale traffic flows. Instead, Du et al. [16] used an irregular convolution kernel that refers to the traffic flow values from the adjacent traffic lines to fill in the values of the hollow points. These methods commonly exploit the CNN architecture that effectively models visual imagery [26–28]. Furthermore, a family of RNN algorithms such as GRU [29] and LSTM [25] was used so that repetitive temporal patterns can be discovered. However, these works did not capture the relation between the flow points on the two-dimensional spaces such as junctions, crossroads, and overpasses. Therefore, these models are susceptible to errors by correlating between irrelevant traffic flows. For instance,

they may confuse overpasses and overlapping roads as crossroads. Furthermore, these works did not address the impact of the external conditions on the traffic flow. Learning the correlation between traffic flows and weather parameters was useful for flow prediction, as presented in [12,30]. However, they overlooked the impact of the substructure around traffic links on traffic flows.

More recently, modeling the traffic flow based on the graph representation of the traffic networks has emerged. The works presented in [18,31–33] combined GNNs (Graph Neural Networks) [34,35] and RNNs to capture the temporal flow transition patterns given adjacency matrices that explicitly reflect the complex interconnections. These methods do not have to unnecessarily deal with the information irrelevant to the traffic flow, such as the hollow points in the visual traffic networks that Du et al. [16] had to consider forcefully.

ST-TrafficNet [33] used Caltrans PeMS (Performance Measurement System) data from around 20 links between intersection points and predicted traffic speed with stacked LSTM using a spatially-aware multi-diffusion convolution block. This PeMS data were from 350 loop detectors at 5 min intervals from 1 January 2017 to 31 May 2017. This model reflects spatial influence through multi-diffusion convolution with forward, backward, and attentive channels.

TGC-LSTM was used by Cui et al. [18] to predict the traffic speed on four connected freeways in the Greater Seattle Area. They used publicly available traffic state data from 323 sensor stations over the entirety of 2015 at 5 min intervals. With their model, traffic speed was predicted with the RMSE (Root Mean Squared Error) as low as 2.1. However, the GNN architecture has to be restructured whenever the traffic networks undergo some changes. This is because TGC-LSTM uses the entire adjacency matrix as an input to the GNN instead of embedding the features of individual traffic links. Upon any change to the traffic networks, we have to re-train from scratch with the newly updated GNN architecture. Furthermore, since TGC-LSTM uses a very large adjacency matrix, both the time and space complexity of modeling the network structure becomes high. However, more importantly, the larger the adjacency matrix is, the more sparse it becomes. Therefore, TGC-LSTM still faces the problem of incorporating unnecessary data such as the hollow points captured in a regular convolution kernel, as discussed in [10]. The shortcomings of these GNN-based approaches motivated us to devise a new method for embedding the characteristics of the traffic network.

We adapt the node embedding techniques introduced by Hamilton et al. [19]. We represent the relationship between links on the traffic network with a feature vector whose size is invariant even when any part of the network structure changes. We analyze how the traffic routes through a link via reachable lowest cost multi-hop paths that are computed with the Floyd–Warshall algorithm [20]. We compute every link's relative impact on other links based on its inbound/outbound traffic flow patterns and its neighbors' collective conditions. We refer to the relative cross-link impact value as Traffic Flow Centrality (TFC). We combine the features related to TFC with the external conditions around each link, such as climate and time information. We use a recurrent neural network algorithm to learn how such a composite feature change over time determines the traffic speed of each link.

Our method does not involve the process of embedding the entire adjacency matrix. Therefore, our solution is more space efficient and can easily handle large-scale traffic networks with thousands of links. Furthermore, it avoids incorporating irrelevant information such as the hollow points that can be present in traffic networks when they are represented with a sparse adjacency matrix or raster graphics [10,18]. Our solution considers the conditions of the remote links beyond the adjacent neighbors. By ruling out irrelevant information and having the broader contextual view, we expect to outperform the works that base their speed prediction myopically on the conditions of the adjacent links.

The advantage of our work, named TFC-LSTM, is summarized in Table 1, which shows the comparison between existing related works we have discussed so far. The "Traffic Network Structure" refers to the usage of the abstract representation of interconnections between links. The "Surrounding Conditions" refer to the consideration of external situations around links such as climate and time information. The "Traffic Flow Reachability Analysis" refers to the process of analyzing

the pattern of traffic flowing in and out of links through reachable paths. The "Centrality Analysis" refers to the usage of the link's relative influence on others. The "Chains of Neighbors" column indicates the consideration of remote neighbors besides the adjacent ones when capturing the substructure around a link.

Table 1. Comparison with other models. TFC, Traffic Flow Centrality.

Prediction Models	Raw Data Usage			Traffic Network Representation			
	Traffic Speed/Volume	Traffic Network Structure	Surrounding Conditions	Invariant Input Feature Vector Size	Traffic Flow Reachability Analysis	Centrality Analysis	Chains of Neighbors
TFC-LSTM	O	O	O	O	O	O	O
TGC-LSTM [18]	O	O	X	X	X	X	X
ST-3DNET [10]	O	O	X	X	X	X	X
DST-ICRL [16]	O	O	O	X	X	X	X
IBCM-DL [17]	O	X	X	X	X	X	X
DBN [15]	O	X	X	X	X	X	X
LSTM [13]	O	X	X	X	X	X	X
GRU [14]	O	X	X	X	X	X	X
DNN [12,22]	O	X	O	X	X	X	X
FNN [8,9]	O	O	X	X	X	O	X
K-NN [6,7]	O	O	X	X	O	X	X
SVR [4,5]	O	O	X	X	X	X	X
ARIMA [1–3]	O	X	X	X	X	X	X

"X" means yes, and "O" means no.

3. Methodology

We outline the overall procedure for predicting the link's speed on the traffic networks, as shown in Figure 1. Given raw data such as adjacency, distance, and speed in a data tensor, we compute reachability information such as hop count, distance, and cost of traffic flowing between a pair of source and destination links through a chain of neighboring links. Paired with other external features such as weather conditions, time of day, and day of the week, we expect that encoding the dynamics of the temporal traffic flows on the neighboring links would significantly improve the prediction of links' speed. How we generate the embedding of complex context-aware spatio-temporal features is explained in greater detail in Section 3.1. We model the correlation between the input feature and each link speed with a Long Short-Term Memory (LSTM) algorithm. We use 512 perceptrons in the hidden layer, ReLU for the activation function [36,37], and Adam for the optimizer [38,39].

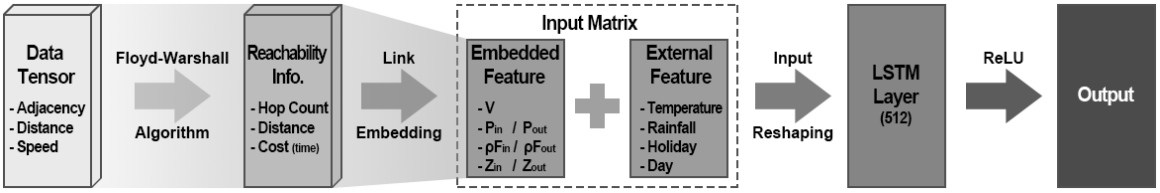


Figure 1. The overall procedure for traffic link embedding and training for traffic speed prediction.

3.1. Link Embedding

Feature representation of a link is done in the following three steps: First, we abstract the traffic system as a link adjacency matrix A. A traffic network illustrated in Figure 2A is abstracted further as a directed graph structure, as shown in Figure 2B, with every directional link being uniquely identified. Figure 3A shows the link adjacency matrix for the sample traffic system in Figure 2A. $A_{l_i l_j}$ denotes whether traffic can flow from link l_i to link l_j. For instance, according to the example in Figure 2A,

$A_{l_2 l_3} = 1$ since traffic can flow from l_2 to l_3. Otherwise, $A_{l_2 l_3}$ equals 0. We exclude the links through which no traffic can reach any other parts of the traffic system. For instance, $l1$ and $l16$ are disconnected from the rest of the traffic system. Figure 3B illustrates a matrix of the distance between every pair of adjacent links. Given $A_{l_i l_j}$, Figure 3C shows the speed of the traffic flowing from l_i to adjacent link l_j.

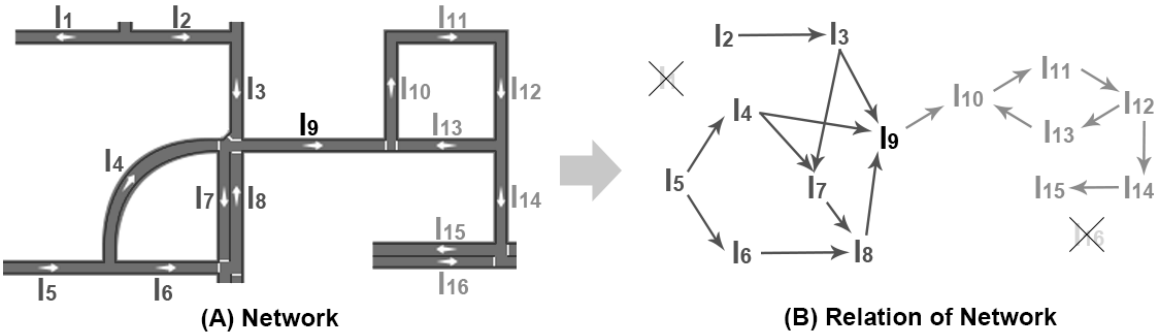


Figure 2. Relation and bound of the network.

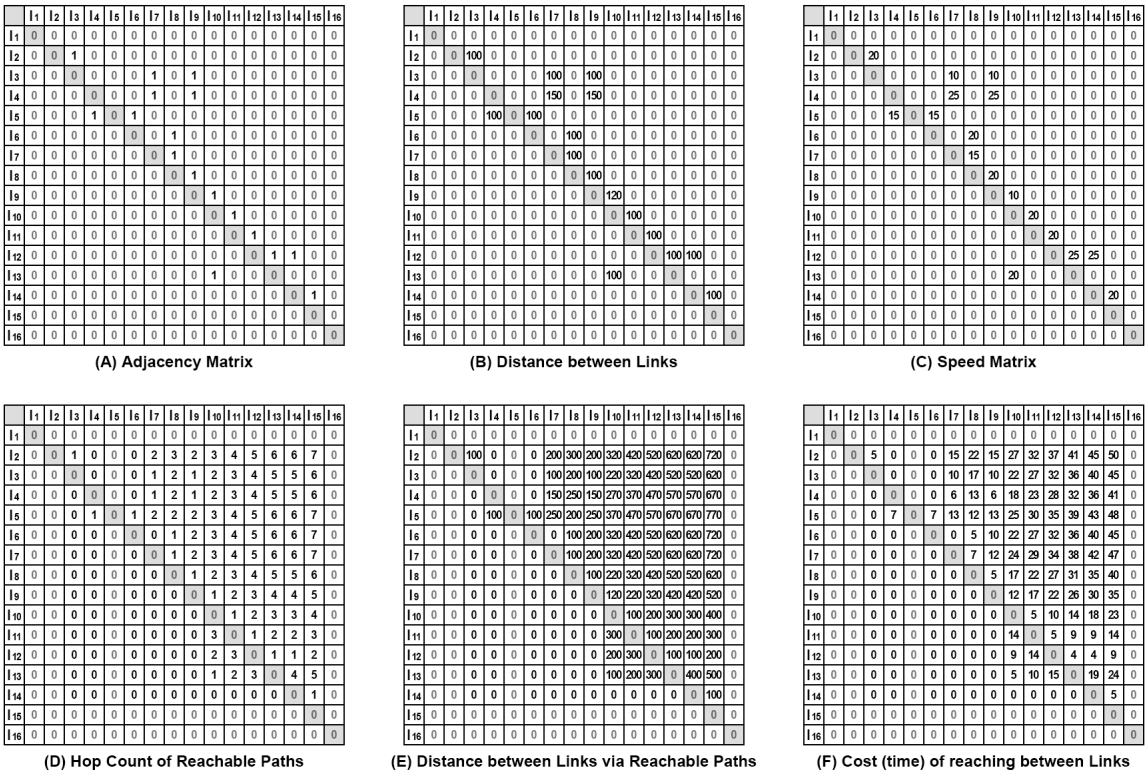


Figure 3. Matrices representing adjacency, distance, and information about reachable paths between links.

Second, we conduct a traffic flow reachability analysis. For every link l, we extract all inbound multi-hop paths through which traffic traverses towards l. Oppositely, we compute all outbound multi-hop paths through which the traffic originated from l diffuses. Given $A_{l_i l_j}$, we run the Floyd–Warshall algorithm [20] to obtain the minimum time it takes to travel from every source link l_i to all other destination links l_j, as shown in Figure 3F. We generate a matrix of hop counts and minimum distances from every source link l_i to all other destination link l_j, as shown in Figure 3D,E, respectively. For instance, the traffic on l_5 (at Hop 3) takes l_6 (at Hop 2) to reach l_9 at time t_2 in Figure 4B as opposed to taking l_4 (at Hop 1) at time t_1 in Figure 4A due to the heavy congestion on l_4. After we obtain the paths, we discard the links through which traffic cannot reach the source link before the beginning of the next time window. For example, if the traffic on l_9 is too slow to reach the remote link

l_{15} by the end of the current time window, then we discard l_{15} from the reachable outbound path from l_9, as shown in Figure 5.

$$P_{in} = \sum_{i \in R}^{R} C_{in} \quad , \quad P_{out} = \sum_{i \in R}^{R} C_{out} \tag{1}$$

$$\rho F_{in} = \sum_{i \in R}^{R} \frac{v_{in_i}}{f_{in_i} d_{in_i}} \quad , \quad \rho F_{out} = \sum_{i \in R}^{R} \frac{v_{out_i}}{f_{out_i} d_{out_i}} \tag{2}$$

$$L_{x_{in}} = \frac{\rho F_{in}}{\sum_{i \in R}^{R} \rho F_{in}} \quad , \quad L_{x_{out}} = \frac{\rho F_{out}}{\sum_{i \in R}^{R} \rho F_{out}} \tag{3}$$

$$Z_{in} = \ln\left(\sum_{n \in N_{in}}^{N_{in}} \frac{L_{n_{in}}}{e^{hop}}\right), \; Z_{out} = \ln\left(\sum_{n \in N_{out}}^{N_{out}} \frac{L_{n_{out}}}{e^{hop}}\right) \tag{4}$$

$$Z_{in} = \frac{Z_{n_{in}}}{\sum_{n \in N}^{N} Z_{n_{in}}}, \; Z_{out} = \frac{Z_{n_{out}}}{\sum_{n \in N}^{N} Z_{n_{out}}} \tag{5}$$

$$|Z_{in} - Z'_{in}| < \epsilon \, , \; |Z_{out} - Z'_{out}| < \epsilon \tag{6}$$

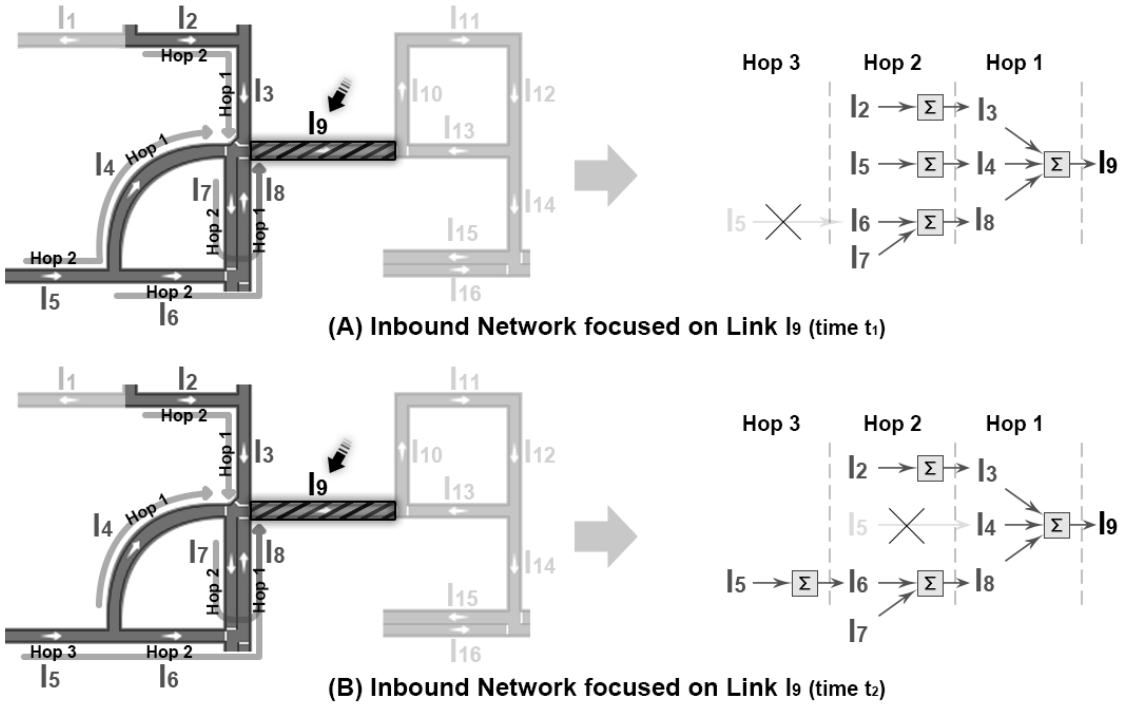


Figure 4. Extraction of inbound reachable paths and the computation of the inbound Z value for I_9.

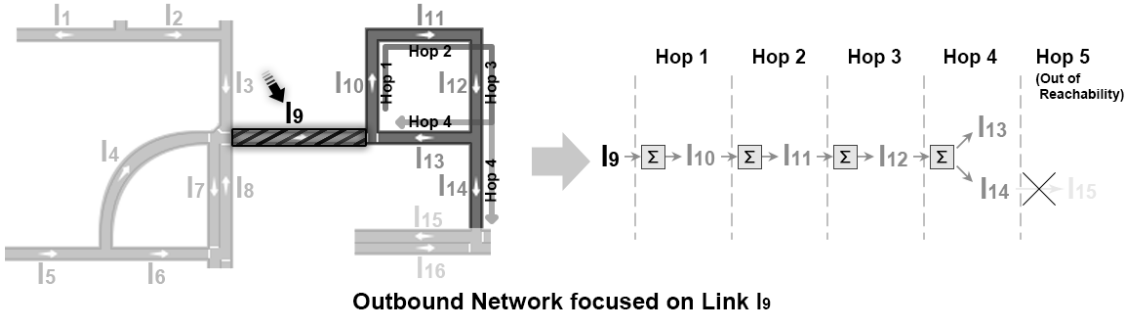
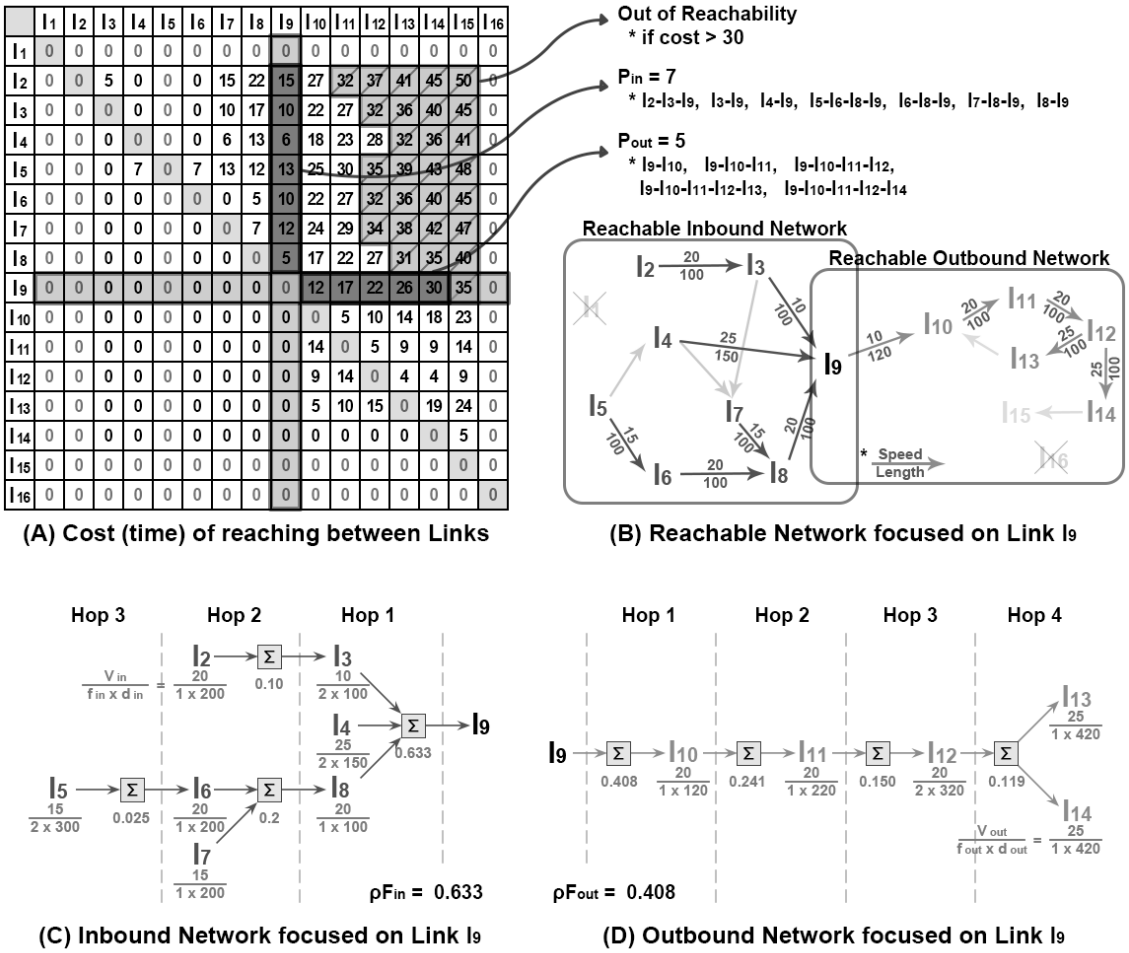


Figure 5. Extraction of outbound reachable paths and the computation of the outbound Z value for I_9.

In the final step, we compute the Z value for every link, which we refer to as the traffic flow centrality. Given the matrix of minimum time to travel from source link l_i to destination link l_j (Figure 6A), we count the number of inbound and outbound reachable paths (R), (P_{in} and P_{out})for every link using Equation (1), as illustrated in Figure 6B. We also compute the speed and distance between every pair of adjacent links on the reachable paths. Equation (2) defines (ρF) as the weighted sum of the average traffic speed (v) on every link i on the multi-hop reachable paths. The weight of each intermediate link i is the inverse of the product between the fanout f and the distance d to i, where f specifically represents the number of alternate paths on a junction. The weight represents the impact on a given link the current traffic is either destined to or stemmed from. We expect the impact to be sensitive to the f and d values. For instance, we can capture the circumstance where the traffic on links with higher f and d values are less likely to move towards a target link l than the traffic on a link with lower f and d values. This is because traffic on the link with higher f and d values is more likely to veer away by taking different turns on the junctions or halt the transition at any point on the path. As an example, computing the inbound and the outbound ρF values for the link l_9 is illustrated in Figure 6C,D. The aggregation steps above are to account for the traffic flow dynamics around every link.



	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}	l_{11}	l_{12}	l_{13}	l_{14}	l_{15}	l_{16}
l_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l_2	0	0	5	0	0	0	15	22	15	27	32	37	41	45	50	0
l_3	0	0	0	0	0	0	10	17	10	22	27	32	36	40	45	0
l_4	0	0	0	0	0	0	6	13	6	18	23	28	32	36	41	0
l_5	0	0	0	7	0	7	13	12	13	25	30	35	39	43	48	0
l_6	0	0	0	0	0	0	0	5	10	22	27	32	36	40	45	0
l_7	0	0	0	0	0	0	0	7	12	24	29	34	38	42	47	0
l_8	0	0	0	0	0	0	0	0	5	17	22	27	31	35	40	0
l_9	0	0	0	0	0	0	0	0	0	12	17	22	26	30	35	0
l_{10}	0	0	0	0	0	0	0	0	0	0	5	10	14	18	23	0
l_{11}	0	0	0	0	0	0	0	0	0	14	0	5	9	9	14	0
l_{12}	0	0	0	0	0	0	0	0	0	9	14	0	4	4	9	0
l_{13}	0	0	0	0	0	0	0	0	0	5	10	15	0	19	24	0
l_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0
l_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6. An example of aggregating the traffic conditions of the neighboring links on reachable paths.

Note that the links adjacent to each other are also inter-dependent on each other with regards to computing their Z values. Due to the inter-dependence, we have to compute Equations (4) and (5) iteratively until the condition on Equation (6) holds. We obtain a converged Z value when Equation (6) is satisfied. We re-scale the ρF values to L_x through normalization as specified in Equation (3). Suppose we have a set of adjacent inbound and outbound neighbors N_{in} and N_{out}, respectively, for a given link. Then, we take the sums of L_x/e^{hop} of every neighbor in N_{in} and N_{out}. The division by e^{hop} reflects that the impact of a link's L_x value on its immediate neighbor is inversely proportional to the hop distance between them. We take the natural logarithmic function on the sums as shown in Equation (4) and normalize the value as defined in Equation (5). Whenever Equation (4) repeats, L_n is substituted by Z' obtained in the previous iteration. The ϵ value in Equation (6) is for determining the converged Z. We empirically set the ϵ value that leads to the most accurate link speed prediction. The iterative computation of the Z value for every link in a traffic network is illustrated in Figure 7.

The traffic flow centrality is to capture the inter-link relationship, and we expect it to be one of the key factors for accurately predicting traffic speed. Intuitively, it is highly probable that a link would experience traffic swarming in and causing congestion, especially when a large portion of its immediate neighbors also experience high inbound traffic through the reachable paths. Furthermore, a link with several surrounding links that spread out traffic quickly is likely to disseminate its traffic more easily.

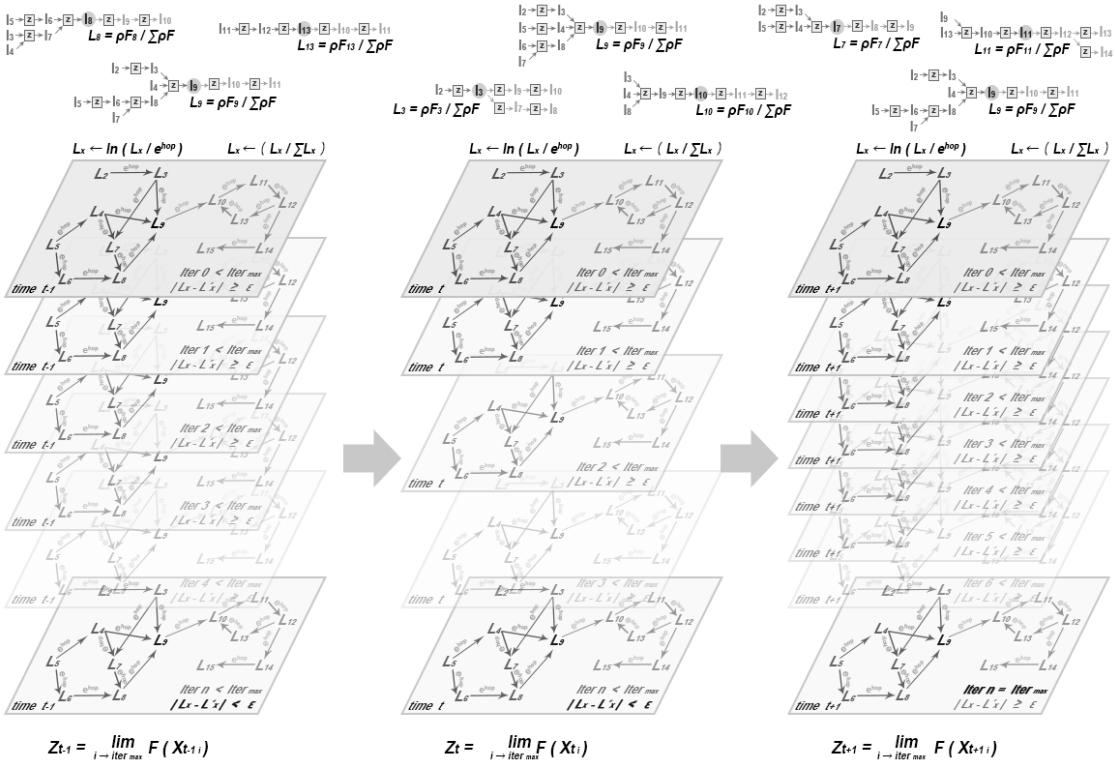


Figure 7. Iterative computation of traffic flow centrality for every link in the traffic network.

We embed the 7 features, (V, P_{in}, P_{out}, ρF_{in}, ρF_{out}, Z_{in}, Z_{out}), into a vector for every link, where V is the traffic speed. We make this link embedding more context-aware by simply concatenating an additional vector of external conditions surrounding the link. The external conditions include temperature, precipitation, time of day, day of the week, and an indication of whether a given day is a public holiday. The steps for generating the final input matrix we feed into a neural network for speed prediction is illustrated in Figure 8.

No matter how many adjacent neighbors a link has, the length of the input vector remains invariant, thus making our solution resilient to changes such as the addition or deletion of neighboring links. We do not take the entire adjacency matrix as an input to the prediction engine based on recurrent neural networks. Therefore, our approach becomes space-efficient. Our input vectors contain only the essential information instead of the initial raw adjacency matrix that is sparse. With every link expressed with highly distinct features, we expect it to yield better prediction results. Note that the input vectors are computed at every time window. In the following section, we introduce the method for modeling the transition of the features over time.

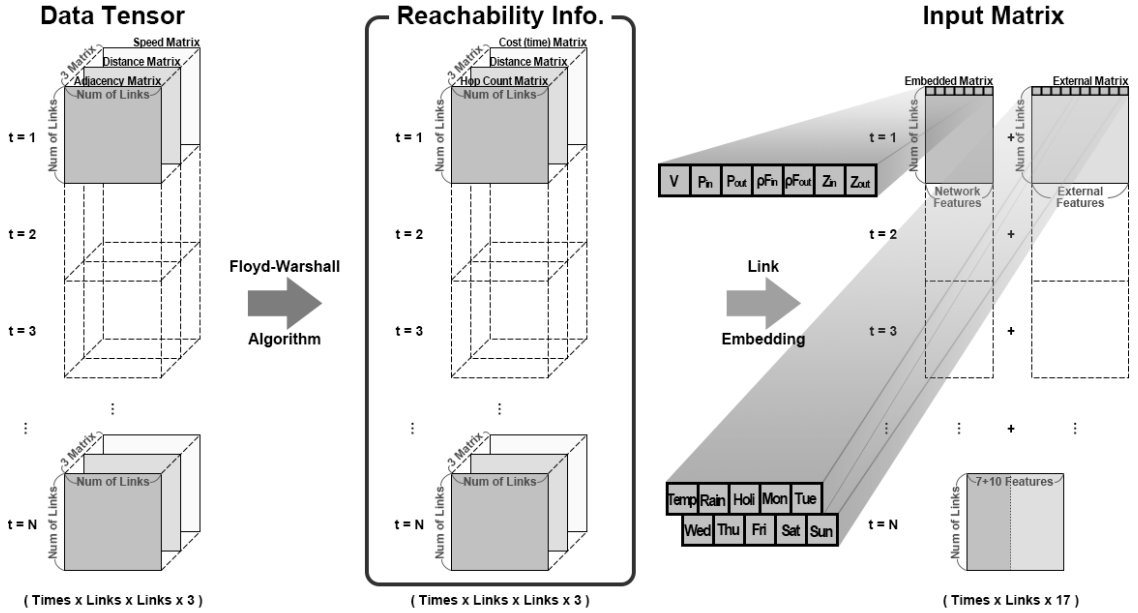


Figure 8. Generation of the input tensor reflecting the context-aware traffic flow centrality of every link.

3.2. Modeling the Temporal Patterns with Recurrent Neural Networks

Given the input matrix generated at a time window, we predict the traffic speed of every link in the next time window using recurrent neural networks, such as GRU and LSTM. With the hidden layers, we can model a complex non-linear relationship between the input and the output values. Specifically, the example of feeding in the input features of every link and predicting its speed through an LSTM network is illustrated in Figure 9. The states summarized in the previous time window are fed into the block of hidden layers at the subsequent time window. Besides the previous states, we feed in the updated input feature vectors of every link. By training this network, we can model the spatial information's temporal transition, such as the transition of the traffic system's structure, traffic flow dynamics, and external conditions. We also make the correlation between the traffic speed of each link and the temporal transition of the most comprehensive set of features to date.

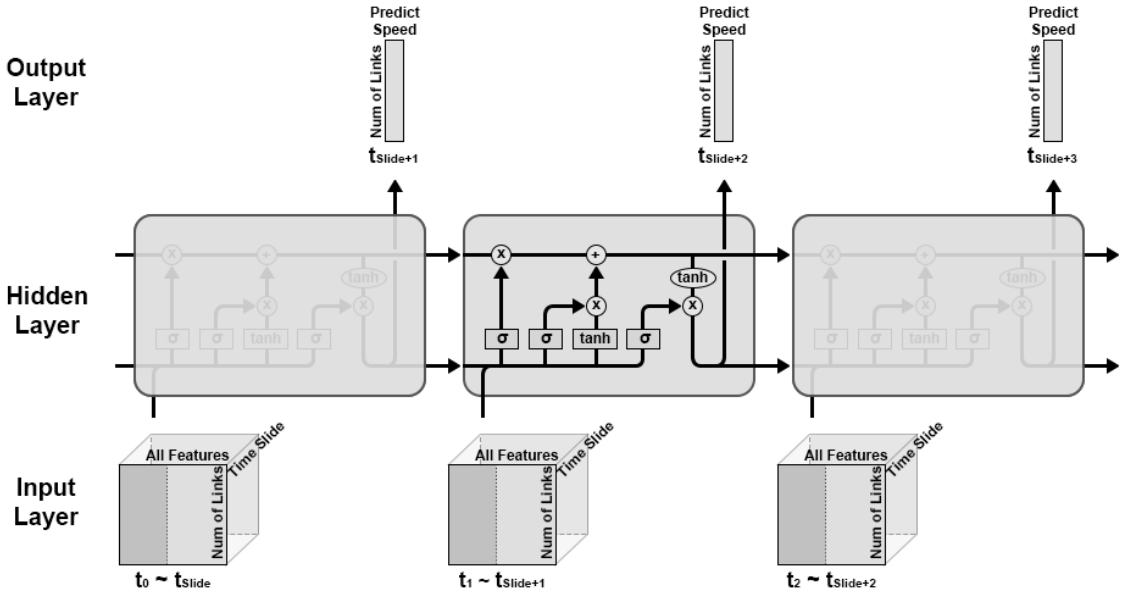


Figure 9. Using LSTM for modeling the correlation between the temporal patterns and the speed prediction at every time window.

4. Evaluation

In this section, we evaluate our approach using the data from TOPIS (Seoul Traffic Operation and Information service) https://topis.seoul.go.kr/, which contain hourly average speed information for the 4670 major traffic links in the Seoul metropolitan area. We obtained data at one hour intervals for 8760 h worth of data from 00:00 on 1 January 2018 to 23:00 on 31 December 2018. Temperature and precipitation information of each link was obtained from the nearest weather station. We retrieved the climate readings from KMA's https://data.kma.go.kr/ AWS(Automatic Weather System). We took 60% of the data as a training set, 20% as a validation set, and the rest as the test set. The attributes of the dataset used in this paper are listed in Table 2 with basic statistics, units, measurement intervals, and data types.

Table 2. Data attributes.

Data	Speed	Temperature	Precipitation	Day	Holiday
Mean	28.17	12.88	0.14	-	-
Median	25.62	13.7	0	-	-
Max	110.00	40.5	76	1	1
Min	0.74	−27.1	0	0	0
Unit	km/h	°C	mm	-	-
Time Interval	1 h	1 h	1 h	1 day	1 day
Type	float	float	float	binary	binary

We conducted our experiments on NVIDIA DGX-1 with an 80-Core (160 threads) CPU, 8 Tesla V100 GPUs with 32 GB of exclusive memory and 512 GB of RAM. NVIDIA DGX-1 was operated with the Ubuntu 16.04.5 LTS server, and the machine learning tasks were executed through the Docker containers. The machine learning algorithms were implemented with Python (v3.6.9), Tensorflow (v1.15.0), and Keras (v2.3.1) libraries. We used ReLU for the activation function [36,37] and Adam for the optimization function [38,39]. The learning rate was empirically set to 0.001. The early stoppage was configured by setting the patience value to 200 through Keras. To measure the prediction

performance, we employed the metrics as follows: (1) MAE (Mean Absolute Error) (Equations (7)); (2) RMSE (Root Mean Squared Error) (Equation (8)); and (3) MAPE (Mean Absolute Percentage Error) (Equation (9)). y_t and $\hat{y}_t$ are the predicted speed and the actual speed, respectively. n is the number of test cases. The ϵ value for checking the convergence of the Z value, as defined in Equation (6), was empirically set as shown in Table 3. With ϵ = 0.005, we were able to achieve the lowest MAPE.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{7}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{8}$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| * 100 \quad (\%) \tag{9}$$

Table 3. Prediction performance according to the convergence condition (ϵ) setting.

Epsilon	MAE	RMSE	MAPE
0.1	2.60	4.40	10.84
0.05	2.69	4.48	11.16
0.01	2.63	4.41	11.07
0.005	2.50	4.28	10.39
0.001	2.69	4.43	11.15
0.0005	2.59	4.36	10.79
0.0001	2.54	4.32	10.51

4.1. Measurement of Prediction Performance

We measured the prediction performance between various approaches denoted as DNN, TFC-DNN, GRU, TFC-GRU, LSTM, and TFC-LSTM. The prefix TFC- stands for Traffic Flow Centrality, and it represents the following seven novel features that we introduced in Section 3, i.e., V, P_{in}, P_{out}, ρF_{in}, ρF_{out}, Z_{in}, and Z_{out}. We evaluated the effectiveness of the information about external conditions such as climate and date information separately. DNN, GRU, and LSTM are the artificial neural network architectures we employed for machine learning. Table 4 contains the prediction performance of each approach. For each model, we picked the empirically best hyper-parameter settings, such as the number of hidden layers, perceptrons per layer, and the number of time windows for the recurrent neural networks. For double hidden layers, we had 64 and eight perceptrons for the first and the second layer, respectively. For a single hidden layer, we used 512 perceptrons. We cited the representative existing works that fall under the prediction model categories that do not use our techniques. We did not compare the methods that could not deal with the traffic networks that scale to thousands of links. MSE values during the training and validation stage are plotted on the graphs in Figure 10. MSEs converged at epoch = 400.

The consideration of every link's external conditions was effective only when used by LSTM and TFC-LSTM. On the other hand, we observed that using the features related to traffic flow centrality consistently led to improvement over the baseline approaches. However, when both the external conditions and the features related to traffic flow centrality were used with LSTM, i.e., TFC-LSTM, we achieved the lowest MAPE of 10.39.

Table 4. Comparison of prediction performance.

Prediction Models	Usage of Surrounding Condition Information	Hidden Layer Structure (# of Perceptrons per Layer)	# of Time Windows	MAE	RMSE	MAPE
DNN [12,22]	No	Double layers (64-8)	-	4.05	6.25	15.58
	Yes	Double layers (64-8)	-	4.36	6.42	17.36
TFC-DNN	No	Double layers (64-8)	-	3.84	5.98	15.27
	Yes	Double layers (64-8)	-	3.96	6.01	17.02
GRU [14]	No	Single layer (512)	18	3.70	5.51	14.42
	Yes	Single Layer (512)	18	7.52	10.20	31.12
TFC-GRU	No	Single layer (512)	18	2.85	4.60	11.42
	Yes	Single Layer (512)	18	6.20	8.12	22.57
LSTM [13]	No	Single layer (512)	18	2.88	4.91	11.89
	Yes	Single Layer (512)	18	2.64	4.48	10.87
TFC-LSTM	No	Single layer (512)	18	2.63	4.47	11.38
	Yes	Single layer (512)	18	2.50	4.28	10.39

The DNN-based approaches without the information about the external conditions performed poorly compared to the recurrent neural network models because it does not model the temporal transitions of the link features.

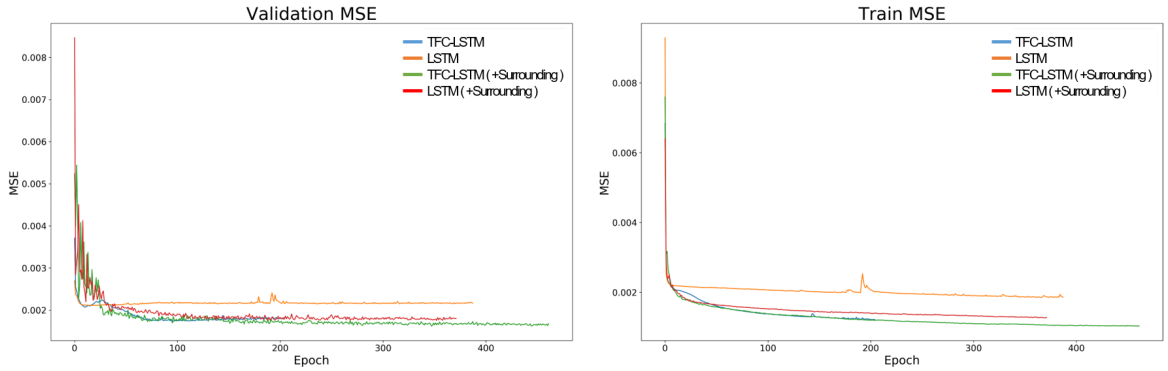


Figure 10. MSEs during training and validation.

4.2. The Effect of Reachable Path Length Cutoff

We noticed that a link in the Seoul traffic system could be reached from any other links within an hour regardless of the distance, even during rush hours with heavy traffic, according to the Floyd–Warshall algorithm we used for retrieving the lowest cost inbound and outbound paths between any OD pairs. It turns out that the Floyd–Warshall algorithm retrieved several unrealistic reachable paths between any OD pairs by ignoring specific restrictions on some of the traffic links. For example, the algorithm generated some routes that included wild U-turns that are not allowed on some roads. As a result, path lengths measured as the number of hops tended to be excessively long. Thus, we were unnecessarily taking into account the state of the links for which it is practically impossible to influence the state of the remote links.

One possible solution to this problem is to have a shorter time window than an hour. However, TOPIS only makes the hourly data available to the public. Therefore, we revised the algorithm to limit the path length instead. When overall traffic flows at the lowest average speed, we limited the reachable inbound and outbound path lengths to 15 hops. On the other hand, for the period when overall traffic flows at the highest average speed, we relaxed the length limit to 45 hops. Furthermore, the link's low speed may be attributed to the congestion on the neighboring links in the vicinity. Thus, reachability from other links to the low-speed link is also limited. Therefore, it is sufficient to consider only the links in proximity. Compared to the period of lowest average traffic speed ("Lowest Speed

Period" in Table 5), the reachability of the traffic from one link to the others is higher during the highest average traffic speed period. Thus, for such a period ("Highest Speed Period" in Table 5), we considered the states of the other links within a wider range. Our simple revision of the algorithm is empirically proven to be effective, as shown in Table 5. It shows the prediction performance for different path length cutoff settings. We observed that our approach performed most effectively with an MAPE of 10.39 when the range of neighboring links to consider was proportional to the traffic's average speed.

Table 5. The effect of reachability path length limit on speed prediction.

Lowest Speed Period	Highest Speed Period	MAE	RMSE	MAPE
15 hops	15 hops	2.61	4.38	10.83
30 hops	30 hops	2.51	4.30	10.48
45 hops	45 hops	2.61	4.40	10.96
15 hops	45 hops	2.50	4.28	10.39
45 hops	15 hops	2.63	4.44	10.98

4.3. The Discussion on Scalability

One of the merits of our approach is the capability to predict the traffic speeds even for a very large-scale traffic network such as the road system of Seoul. The larger the traffic network is, the higher the opportunity is to predict speed accurately. This is because we can consider more conditions around the links that are often overlooked by the existing works. The works that only consider the conditions of adjacent neighbors [18] exhibited a decline in prediction accuracy compared to what was originally reported and performed worse than our approach. This is because their approaches were unableto capture the farther away links' highly probable influence.

Naively feeding in the raw adjacency matrix was the most space inefficient approach [40,41]. We could not even compare the prediction performance with such approaches, as they quickly encountered out-of-memory errors when dealing with Seoul's large-scale road system. Our approach is agnostic of the scale of the network and even the structure change. Regardless of the scale and any changes, we ran the aggregation functions to compute the fixed-length input feature vector concerning the traffic flow centrality. Therefore, we did not need to restructure the neural network architecture upon changes to the traffic system (i.e., addition/deletion of links, change of adjacent links).

However, we dealt with a fragment of the entire national traffic system in South Korea. The traffic networks in Seoul are connected to the systems in other districts such as Gyeonggi Province and Incheon metropolitan area. Due to the fragmented view, we accidentally identified the links that bridge between different traffic networks as dead-ends, as shown in Figure 11. We could not accurately reflect the traffic flow centrality for these dead-end links. The Z_{out} value was zero for these dead-end links because there is no way out. The Z_{in} value is not credible as the inbound traffic from other regions was not accounted for. The inaccurate Z values on the dead-end may negatively impact the neighboring links' Z values.

For this issue, we applied the average Z value of the whole system as the Z value of the dead-end links, which still cannot be viewed as an ideal solution. This motivated us to venture into applying our approach to predicting speed prediction for all the links in the entire national traffic system. By expanding the view of the traffic network, we expect the accuracy to improve further. This is planned to be done soon after we are given integrated traffic information from all Korean regions.

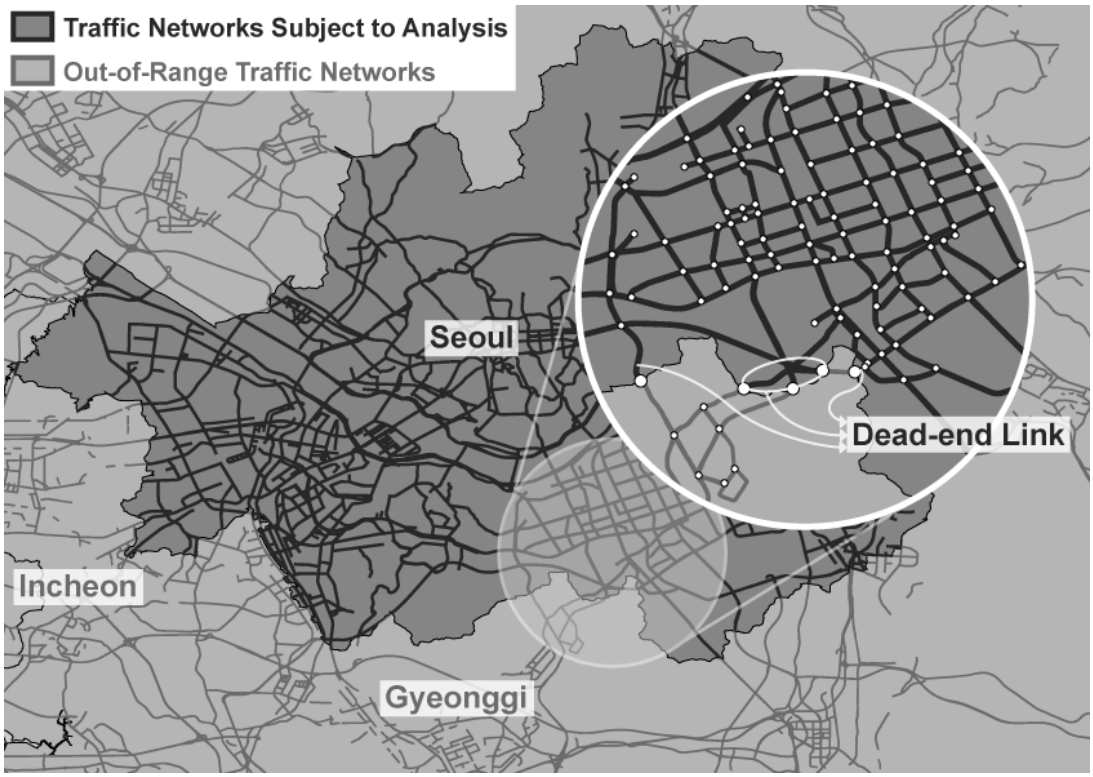


Figure 11. Bridge links between traffic networks subject to analysis and out-of-range traffic network accidentally being recognized as dead-end links.

5. Conclusions

We presented a novel method for describing the dynamic circumstances around any given traffic links. Specifically, by using a new measure called traffic flow centrality, we were able to concisely express the dynamics of the traffic flow in and out of any given link. Through this measure, we can reflect on the inherently complex ways in the interconnected traffic links affecting each other. Combined with the information about the external conditions surrounding the links, e.g., climate and time of day, the new features and their temporal patterns are used for predicting traffic speed. According to the information available from TOPIS (Seoul Traffic Operation and Information service), training with the LSTM algorithm given our comprehensive spatio-temporal features yielded the lowest prediction error with an MAPE of 10.39. Our experiment also shows that our solution is easily applicable to large-scale traffic systems. We could predict the traffic speed on the road network with thousands of links, unlike the existing works without any efficient feature embedding approaches.

As future work, we plan to apply our solution to the nation-wide traffic system.

Author Contributions: Conceptualization, Y.Y.; methodology, C.L. and Y.Y.; software, C.L.; validation, C.L. and Y.Y.; formal analysis, C.L. and Y.Y.; investigation, C.L.; resources, C.L.; data curation, C.L.; writing, original draft preparation, C.L. and Y.Y.; writing, review and editing, C.L. and Y.Y.; visualization, C.L.; supervision, Y.Y.; project administration, Y.Y.; funding acquisition, Y.Y. All authors read and agreed to the published version of the manuscript.

Funding: This research was supported by the Korean Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure, and Transport (Grant 20TLRP-B148970-03), by the Basic Science Research Programs through the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT) (2020R1F1A104826411), by the Ministry of Health & Welfare, Republic of Korea (Grant Number HI19C0542020020), by the Ministry of Trade, Industry & Energy (MOTIE) and the Korea Institute for Advancement of Technology (KIAT), under Grants P0004602 and P0014268 Smart HVAC demonstration support, and by the 2020 Hongik University Research Fund.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

OD Origin and Destination
TFC Traffic Flow Centrality (Z value)

References

1. Williams, B.M.; Hoel, L.A. Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results. *J. Transp. Eng.* **2003**, *129*, 664–672.
2. Kumar, S.V.; Vanajakshi, L. Short-Term Traffic Flow Prediction using Seasonal ARIMA Model with Limited Input Data. *Eur. Transp. Res. Rev.* **2015**, *7*, 21.
3. Chen, C.; Hu, J.; Meng, Q.; Zhang, Y. Short-Time Traffic Flow Prediction with ARIMA-GARCH Model. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 5–9 June 2011; pp. 607–612.
4. Jeong, Y.S.; Byon, Y.J.; Castro-Neto, M.M.; Easa, S.M. Supervised Weighting-Online Learning Algorithm for Short-Term Traffic Flow Prediction. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 1700–1707.
5. Wu, C.H.; Ho, J.M.; Lee, D.T. Travel-Time Prediction with Support Vector Regression. *IEEE Trans. Intell. Transp. Syst.* **2004**, *5*, 276–281.
6. Cai, P.; Wang, Y.; Lu, G.; Chen, P.; Ding, C.; Sun, J. A Spatiotemporal Correlative K-Nearest Neighbor Model for Short-Term Traffic Multistep Forecasting. *Transp. Res. Part C Emerg. Technol.* **2016**, *62*, 21–34.
7. Kim, T.; Kim, H.; Lovell, D.J. Traffic Flow Forecasting: Overcoming Memoryless Property in Nearest Neighbor Non-Parametric Regression. In Proceedings of the 2005 IEEE Intelligent Transportation Systems, 2005, Vienna, Austria, 16 September 2005; pp. 965–969.
8. Yin, H.; Wong, S.; Xu, J.; Wong, C. Urban Traffic Flow Prediction using a Fuzzy-Neural Approach. *Transp. Res. Part C Emerg. Technol.* **2002**, *10*, 85–98.
9. Quek, C.; Pasquier, M.; Lim, B.B.S. POP-TRAFFIC: A Novel Fuzzy Neural Approach to Road Traffic Analysis and Prediction. *IEEE Trans. Intell. Transp. Syst.* **2006**, *7*, 133–146.
10. Guo, S.; Lin, Y.; Li, S.; Chen, Z.; Wan, H. Deep Spatial-Temporal 3D Convolutional Neural Networks for Traffic Data Forecasting. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3913–3926.
11. 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.
12. Koesdwiady, A.; Soua, R.; Karray, F. Improving Traffic Flow Prediction with Weather Information in Connected Cars: A Deep Learning Approach. *IEEE Trans. Veh. Technol.* **2016**, *65*, 9508–9517.
13. Tian, Y.; Pan, L. Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network. In Proceedings of the 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, China, 19–21 December 2015; pp. 153–158.
14. Fu, R.; Zhang, Z.; Li, L. Using LSTM and GRU Neural Network Methods for Traffic Flow Prediction. In Proceedings of the 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, China, 11–13 November 2016; pp. 324–328.
15. Huang, W.; Song, G.; Hong, H.; Xie, K. Deep Architecture for Traffic Flow Prediction: Deep Belief Networks with Multitask Learning. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 2191–2201.
16. Du, B.; Peng, H.; Wang, S.; Bhuiyan, M.Z.A.; Wang, L.; Gong, Q.; Liu, L.; Li, J. Deep Irregular Convolutional Residual LSTM for Urban Traffic Passenger Flows Prediction. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 972–985.
17. Gu, Y.; Lu, W.; Xu, X.; Qin, L.; Shao, Z.; Zhang, H. An Improved Bayesian Combination Model for Short-Term Traffic Prediction with Deep Learning. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 1332–1342.
18. Cui, Z.; Henrickson, K.; Ke, R.; Wang, Y. Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. *IEEE Trans. Intell. Transp. Syst.* **2019**. doi:10.1109/TITS.2019.2950416.

19. Hamilton, W.L.; Ying, R.; Leskovec, J. Representation Learning on Graphs: Methods and Applications. *arXiv* **2017**, arXiv:1709.05584.
20. Floyd, R.W. Algorithm 97: Shortest Path. *Commun. ACM* **1962**, *5*, 345.
21. 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.
22. Kim, D.H.; Hwang, K.Y.; Yoon, Y. Prediction of Traffic Congestion in Seoul by Deep Neural Network. *J. Korea Inst. Intell. Transp. Syst.* **2019**, *18*, 44–57.
23. Zheng, Z.; Yang, Y.; Liu, J.; Dai, H.N.; Zhang, Y. Deep and Embedded Learning Approach for Traffic Flow Prediction in Urban Informatics. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3927–3939.
24. Sun, S.; Wu, H.; Xiang, L. City-wide traffic flow forecasting using a deep convolutional neural network. *Sensors* **2020**, *20*, 421.
25. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780.
26. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
27. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Li, F.-F. Large-Scale Video Classification with Convolutional Neural Networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1725–1732.
28. Ji, S.; Xu, W.; Yang, M.; Yu, K. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *35*, 221–231.
29. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
30. Jia, Y.; Wu, J.; Xu, M. Traffic Flow Prediction with Rainfall Impact using a Deep Learning Method. *J. Adv. Transp.* **2017**, *2017*.
31. Yu, B.; Lee, Y.; Sohn, K. Forecasting Road Traffic Speeds by Considering Area-Wide Spatio-temporal Dependencies based on a Graph Convolutional Neural Network (GCN). *Transp. Res. Part C Emerg. Technol.* **2020**, *114*, 189–204.
32. Ge, L.; Li, S.; Wang, Y.; Chang, F.; Wu, K. Global Spatial-Temporal Graph Convolutional Network for Urban Traffic Speed Prediction. *Appl. Sci.* **2020**, *10*, 1509.
33. Lu, H.; Huang, D.; Song, Y.; Jiang, D.; Zhou, T.; Qin, J. St-trafficnet: A spatial-temporal deep learning network for traffic forecasting. *Electronics* **2020**, *9*, 1474.
34. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80.
35. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
36. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines; In Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel, 21–24 June 2010.
37. Glorot, X.; Bordes, A.; Bengio, Y. Deep Sparse Rectifier Neural Networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
38. Kingma, D.P.; Ba, J. Adam: A method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
39. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. *arXiv* **2016**, arXiv:1609.04747.
40. Yu, B.; Yin, H.; Zhu, Z. Spatio-temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. *arXiv* **2017**, arXiv:1709.04875.
41. Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; Li, H. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Trans. Intell. Transp. Syst.* **2019**, 21, 3848–3858.