



Traffic Graph Convolutional Network for Dynamic Urban Travel Speed Estimation

Huan Ngo¹ · Sabyasachee Mishra¹

Accepted: 28 October 2022 / Published online: 16 December 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

The dynamic urban link travel speed estimation (DU-LSE) problem has been studied extensively with approaches ranging from model to data driven since it benefits multiple applications in transport mobility, especially in dense cities. However, with drawbacks such as heavy assumption in model-driven and not being capable for big city network in data-driven, there has not been a consensus on the most effective method. This study aims to develop a Sequential Three Step framework to solve the DU-LSE problem using only the passively collected taxi trip data. The framework makes use of two deep learning models namely Traffic Graph Convolution (TGCN) and its recurrent variant $TGCN_{lstm}$ to capture both spatial and temporal correlation between road segments. The proposed framework has three advantages over similar approaches: (1) it uses only the affordable taxi data and overcomes the data's incompleteness both in spatial (full GPS trajectory is not available) and temporal (incomplete historic time-series) domain, (2) it is specifically designed to preserve the directionality nature of traffic flow, and (3) it is capable for large networks. The model results and validations suggest the framework can achieve high enough accuracy and will provide valuable mobility data for cities especially those without traffic sensing infrastructure already in place.

Keywords Dynamic Traffic States Estimation · Link-level · Graph Convolution Network · Taxi Trip Data

✉ Sabyasachee Mishra
smishra3@memphis.edu

Huan Ngo
hhngo@memphis.edu

¹ Department of Civil Engineering, University of Memphis, Memphis, TN 38152, USA

1 Introduction

In the twenty-first century, urbanizations are happening in countries around the globe at an extraordinary speed, which is reflected by the increase in percentage of urban population from 39% in 1980 to 58% in 2019 (World Bank 2019). This rapid urbanization will pose new mobility needs and lay more stress in the transportation system especially in cities with old infrastructure that could not keep up with even the current demand. City planner can ask funds for rehabilitation and expansion of the current infrastructure such as opening more lanes, building new roads, or transit lines. However, the cost effectiveness of these investments is quite often questionable since new infrastructure does not necessarily translate into better mobility as shown in the example of the Braess Paradox (Frank 1981). On the other hand, there has been rising attention in improving urban mobility through Intelligent Transportation System (ITS), which leverages recent state-of-the-art technologies to increase the effectiveness of the system without major investment in the current infrastructure.

In the United States, the States' Department of Transportation (DOT) have already implemented many applications of ITS such as Ramp Metering, which limits the number of vehicle entering highway during peak hour to avoid a costly congestion, or Traffic Signal Coordination, which synchronizes multiple adjacent intersections to enhance a selected directional flow (DOT 2019). One main challenge that has been consistently debated in ITS is traffic state estimation (TSE), which is the process of inferring traffic state variables (e.g., flow, travel time, density, etc..) with partially observed traffic data (Seo et al. 2017). However, studies in TSE vary greatly in the scope of estimation. In spatial scope, certain studies estimate a selected set of road segments such as major roadways (e.g., highways) and/or those implemented with traffic volume sensors (e.g., Inductive Loop). In temporal scope, papers are more focused on short-term prediction (Ermagan and Levinson 2018; Thapa et al. 2022) since the estimation is more reliable. The research area of estimating all road segments in a dense urban network at every time of the day is new and unexplored. To this end, we introduce the Dynamic Urban Link Travel Speed Estimation (DU-LSE) problem which specifically aims at computing link travel speed for every link within a network, at different time of the day and day of the week. The outcome of DU-LSE can be beneficial to a wide variety of application in ITS such as monitoring traffic jam, estimating time of arrival, route planning (Kumar et al. 2019; Nantes et al. 2016; Papageorgiou et al. 2003; Seo et al. 2017; Xu et al. 2020), and even for the emerging technology of autonomous vehicle operations (Fountoulakis et al. 2017; Khan et al. 2017). There are private companies such as INRIX, HERE, or TOMTOM offering traffic estimation services but there are three main concerns for city planner who wishes to adopt this method. First, the estimation coverage of these services may be limited to major segments of the road network such as interstate and in urban mobility, knowledge of both major and minor segments (i.e., Central Business District) is significantly more beneficial. Second, since these services require data collected either from probe vehicle travel program or stationary traffic sensor, cities

that do not have these infrastructures simply cannot utilize this method. Third, estimation for commercial companies is still not as accurate and reliable since there are multiple observations of the original data, and it is not clear on the sampling method. Therefore, there should be an independent alternative framework to compare such information.

DU-LSE has been studied extensively with methods ranging from model-driven to data-driven approaches (Seo et al. 2017). Conventional model-driven approach relies on theoretical principles represented by mathematical formulation to describe the physical traffic flow. On the other hand, data-driven approach such as deep learning relies on a vast amount of data and learns its hidden pattern through the optimization of model's weights and biases. This approach has recently gained traction due to two main factors. *First*, there is an unprecedented growth rate in the amount of data generated even just through passive daily actions. A study in 2013 found that 90% of the world data at that time was created during 2011–2012 alone (Ralph 2013). This abundant source of data functions as a fuel to improve the accuracy of deep learning models. *Second*, continual advancement in computing power has paved the way for processing these large data such as IBM's Infosphere processing at a rate up to 120,000 GPS points per second (Biem et al. 2010). However, in the ITS field and especially in solving DU-LSE, studies using data-driven approach often require data from sensing infrastructure such as inductive loop detectors, license plate recognition devices, or 360° cameras. Cities that wish to take advantage of these studies for their ITS system either need to have these infrastructures already in place or invest in a new sensing infrastructure which can be costly. This poses a problem for cities with emerging population and economy especially those in developing countries. Thus, the use of data that is a byproduct of daily activities and publicly available is desirable. In the case of urban mobility data, taxi trip dataset has great potentials because not only it meets all these criteria but also is abundantly available, especially in dense urban areas. For example, as a result of the Open Data Law signed into effect as of 2012, the New York City Taxi & Limousine Commission (NYC-TLC) has released an astonishing record of 1.1 billion taxi trips from 2009 to 2015 (City of New York 2019). Due to its enormous size, this dataset is perfectly suitable to fuel a deep learning model aiming to solve the DU-LSE problem. Furthermore, the taxi data has extensive both spatial and temporal coverage, which is demonstrated later in Sect. 4 case study.

Although there have been several papers devoted to DU-LSE (Yu et al. 2019; Wu et al. 2015; Liu et al. 2019; Sekuła et al. 2018; Zhan et al. 2013), the literature has not yet reached a consensus on the most effective method because of drawbacks in the methodology such as heavy assumption in model-driven and lack of scalability in data-driven approaches. We shall present a more extensive review of the literature on DU-LSE and identify these drawbacks in Sect. 2. Therefore, this objective of this paper is to *develop a sequential three step framework that leverages a single dataset of taxi trip to estimate historical complete network link travel speed, disaggregated by time of the day.*

The remainder of the paper is organized as follows: Sect. 2 reviews the related literature in the domain of deep learning model in ITS and specifically, the DU-LSE problem. In Sect. 3, we present in detail each step in the sequential three step framework.

This framework is then applied on the New York City Taxi dataset in Sect. 4. Section 5 discusses the evaluation validation of each step in the framework and finally, Sect. 6 concludes the paper with the discussion on findings, model performances, and avenues of future research.

2 Literature Review

The number of studies using deep learning model in ITS is increasing with applications ranging from ridesharing services (Geng et al. 2019; Ke et al. 2017; Yao et al. 2018), bikesharing services (Lin et al. 2018), to car parking demand prediction (Yang et al. 2019). These studies usually utilize a combination of two methods to capture both spatial and temporal relation. For spatial relations, studies utilize variations of Convolutional Neural Network (CNN) which have already achieved tremendous success in the field of image recognition and video classification. The main idea of CNN is to aggregate information of pixels located inside a pre-defined kernel filter and this filter is then transported through the rows and columns of the image's pixel to learn and identify common patterns. However, CNN has difficulty in implementation for road network. Unlike image dataset, which is an Euclidean type data, road network is a graph structure data and there is no notion of direction but only notion of node connectivity. One possible solution is using Graph Convolutional Network proposed by Kipf and Welling (2017). The approach's main idea is that a host node would gather information from its neighbor one "hop" away from itself. The procedure can be repeated multiple times to reach to further neighbors. One thing to remind from this study is it is node-based, which means only node information can be processed whereas information of the links connecting these nodes are ignored.

For temporal relation, variations of Recurrent Neural Network (RNN) especially those in form of Long-Short Term Memory (LSTM) are utilized. The main idea of RNN is, it would take both the current and previous observations as input and the operation is repeated at each state of time and hence the name "Recurrent". However, RNN suffers from the exploding or vanishing gradient problem where changes of model's weights and biases during training are either too small or too big that it could not achieve convergence in the loss function. LSTM addresses this problem by introducing an internal state value at which the gradient flow is uninterrupted and thus avoids the exploding/vanishing gradient problem (Hochreiter and Schmidhuber 1997). This approach has been the state-of-the-art model for capturing temporal relation in situation such as bike-sharing and car parking demand prediction (Lin et al. 2018; Yang et al. 2019). However, most studies using LSTM rely on a complete historic time-series dataset to effectively train the model. Traditionally, LSTM was often implemented in a local fashion where a node would only look back at its historical data and ignores its neighbor's, but more recent papers are starting to embed connectivity into the formulation and promote spatial message passing. For an in-depth review of deep learning model in ITS, we recommend the survey by Wang et al. (2019) where the authors goes through various techniques and applications. One

main take away from this survey is deep learning lack the interpretability power if not formulated appropriately.

In the DU-LSE problem, the common practice for achieving the link-level travel speed often involves a theoretical four step planning process, which are (1) trip generation, (2) trip distribution, (3) modal split, and (4) traffic assignment (Sheffi 1975). However, the four-step planning process requires the planner to issue a Household Travel Survey, which is costly and conducted approximately once every 10 years thus results in a low reliability when being implemented in later period of the collecting cycle. In addition, the process relies on stringent assumptions such as Wardrop User Equilibrium that do not necessarily hold true in the real world (Yu et al. 2017). More recent research can be categorized into either model-based or data-driven approach. In model-based, (Yeon et al. 2008) used probabilistic breakdown for freeway segments and Discrete Time Markov Chain to estimate travel time with data from microwave sensors and CCTV on US202 in Philadelphia, PA. With New York taxi data, (Zhan et al. 2013) proposed a two steps framework where route choice is first estimated by multinomial logit model and travel time is calculated by an optimization model minimizing the expected and observed path travel time. Another high-performance algorithm was introduced by (Wu et al. 2015) where the author used convex optimization coupled with dimensionality reduction scheme and projected gradient algorithm to estimate traffic. The data is a fusion between vehicle count via sensors and cellular network along I-210 region of Los Angeles. Other notable approach in model-based are Tucker decomposition-based imputation (Tan et al. 2013) for PeMS data in Sacramento County and maximum likelihood (Jenelius and Koutsopoulos 2013) for GPS probes in Stockholm, Sweden. In data-driven approach, early work includes a three-layer neural network for low-pooling frequencies probe vehicle data (Zheng and Van Zuylen 2013), denoising stacked autoencoders for Caltrans PeMS (Duan et al. 2016), neural network with linear and hyperbolic layers to capture sharp non-linearity of traffic flow in case of special event such as a Chicago Football Game (Polson and Sokolov 2017), and (Sekula et al. 2018)'s neural network with a profiling model for ATR station and vehicle probe data in Maryland. A more contemporary approach with high performance is the 3D-TGCN by (Yu et al. 2019). The model makes use of Graph Convolutional Network and Dynamic Time Wrapping, and it is applied to the PeMSD7 (2012) and PEMS-BAY (2017) data from Caltrans. One major contribution of this method is DTW introduces less training parameters compared to RNN-based model and the training process is more efficient.

2.1 Research gaps

After reviewing related literature, we identify four research gaps as follows:

1. *First*, most studies rely on well-established and dense dataset that provide a complete historic time-series collected from traffic sensors such as Inductive Loop Detector or License Plate Recognition (Cui et al. 2019; Diao et al. 2019; Wu et al. 2015; Yu et al. 2019; Zhu et al. 2018), floating probe vehicle GPS data (Cui et al. 2019; Yu et al. 2017; Zhu et al. 2018), mobile phone data (Wu et al. 2015; Zhu

- et al. 2018) or even a combination of all of it. However, these datasets require a supporting infrastructure already being implemented in the first place (e.g., sensors) and thus is not applicable to city that does not have such hardware or software system for the entire network. Even if a city decides to implement new sensing facilities, not only it would cost more but also pose a question of where to locate these devices. Other data sources such as GPS or mobile data is not readily available to most researchers and pose concerns about personal security.
2. *Second*, studies using variation of Graph Convolutional Network (Cui et al. 2019; Diao et al. 2019; Kawasaki et al. 2019; Kipf and Welling 2017) are mostly node-based and link information (i.e., road length and number of lanes) are ignored. This also results in an undirected graph where there is only a notion of two nodes being connected by a link but no notion of direction. Therefore, these studies have limitation in capturing the directional flow nature which is inherent in traffic behavior. An example would be, the same highway connecting the suburb and the downtown area, north-bound traffic would differ greatly compared to south-bound traffic at a specific point of time. In addition, the north and south-bound road are represented as two separate links connecting the same origin–destination pair, which cannot be reflected in an undirected graph commonly used in GCN.
 3. *Third*, studies aiming to capture temporal relation by using variations of LSTM (Cui et al. 2019; Lin et al. 2018; Liu et al. 2019) only apply the technique locally, which is a road segment will only look at its own historical data and not its neighbor.
 4. *Finally*, most papers only consider network with moderate size (i.e., up to 500 links) and aim to capture only part of the city transportation network (i.e., highway) especially at segments where traffic sensors are located. In dense urban area, traffic estimation for all links is exponentially more useful than that of for selected highway segments.

2.2 Paper Contribution

1. First, we use only the publicly available Taxi Trip data, which is a byproduct of daily activities and requires minimal investment for City Planners who wish to implement this framework. Taxi data is different than that of probe vehicle data or location-based services data for which the city needs to purchase the data. But for taxi data, many cities have memorandum of understanding with the taxi companies to share the data if they would like to operate in the city with assurance in user privacy. However, the dataset has two main disadvantages that prevents it from being widely used in the DU-LSE field. These are (i) each trip does not have full GPS traces but only the pickup and drop-off coordinates and even these data are completely random, sparse, and far between each other in a network and (ii) no complete historic time-series are provided. The sequential three step framework presented in this paper helps to address this problem of incomplete information so that city planner can use this readily available data for their task of determining DU-LSE. To the best of our knowledge, this is the first paper to

- estimate network-wide travel speed throughout the day with only one single taxi trip data.
2. Second, our paper captures the directional flow, which is to estimate travel speed on both directions of a road segment. This is challenging because conventional GCN works with undirected graph and there is only notions of node-connectivity and neighbor's message passing. We also want to preserve directional flow nature of traffic as explained in the following illustration. At an intersection, there are two traffic flows going in and out of the intersection in the East direction. These two traffic flows can then past message about each other whereas other directions such as North, West, and South are not related. This notion applies to the remaining directions and traffic flow as well. To this end, our paper contributes Traffic Graph Convolutional Network to capture this directional flow nature by using a node-link embedding technique in conjunction with a modified directional adjacency matrix.
 3. Third, for capturing temporal relation, our paper introduces $TGCN_{lstm}$ along with an appropriate model architecture to allow the node to look back at not only of its historical data but also its neighbor's. $TGCN_{lstm}$ makes uses of the core idea from TGCN, such as modified directional adjacency matrix, and LSTM architecture (Hochreiter and Schmidhuber 1997).
 4. Fourth, this paper makes use weight sharing to reduce training parameters and increase training efficiency. At a lower level of TGCN, each node in the network has its own neural network of which computational graph is created by branching from the host node out to its neighbor. By repeating this operation multiple times, the host node can gather information multiple "hops" away. If two nodes i, j are connected together via a real-world road segment, then only one weight w_{ij} is assigned to this pair and w_{ij} is shared across multiple computational graphs. We will explain this notion further in Sect. 3.8.5 of Computational Graph. This facilitates our case study to estimate dynamic urban link travel speed for large network within reasonable computational time. The number of training parameters is independent of the size of the taxi dataset, but the model accuracy benefits greatly from this taxi dataset size.

3 Methodology

3.1 Problem Definition

We first describe the problem as follows. The framework takes input of a taxi trip dataset where each taxi $i \in I$ contains the information about (1) the pickup and dropoff coordinates, pickup and dropoff time, and travel distance. This information is represented as a tuple:

$$\{x_i^{start}, y_i^{start}, x_i^{end}, y_i^{end}, t_i^{start}, t_i^{end}, d\}, \forall i \in I$$

The output of the model is the travel speed of every link in the network at every time of the day as follows: $y_{at}, \forall a \in A, t \in T$ where A is the set of links and T is the

set of Time period. All notations for this paper are summarized and presented in Appendix A. In addition, physical meaning of every notation will be mentioned and explained in text after it is introduced.

3.2 Overall Framework

In order to solve the DU-LSE using taxi trip data, we use a Sequential Three Step (S3S) framework which consists of (1) Path Choice Prediction, (2) Partial Link Travel Time Prediction, and (3) Dynamic Urban Link Travel Speed Estimation. The overall interactions between these steps are as follows. Step 1 takes the raw input which is taxi trip data and produces an output of predicted path choice for all taxi trips. This output is fed as an input to step 2 which then produces an output of partial link travel time. Step 3 then takes the step 2 output as input and produces the ultimate result which is the DU-LSE. Our study uses link-level travel speed as representation of traffic state instead of travel time for reasons discussed later in step 3. It's important to note that one can readily compute link travel time by dividing link length with travel speed. A schematic view of S3S framework's architecture and details of each step are provided in Fig. 1. For the analysis of step 1 and 2, we first subset the taxi trip from the master dataset by a 30-min time window (i.e., 01/01/2014 06:00–06:30). Steps 1 and 2 are then independently processed for each time window and repeated until every time windows within the analysis period (i.e.,

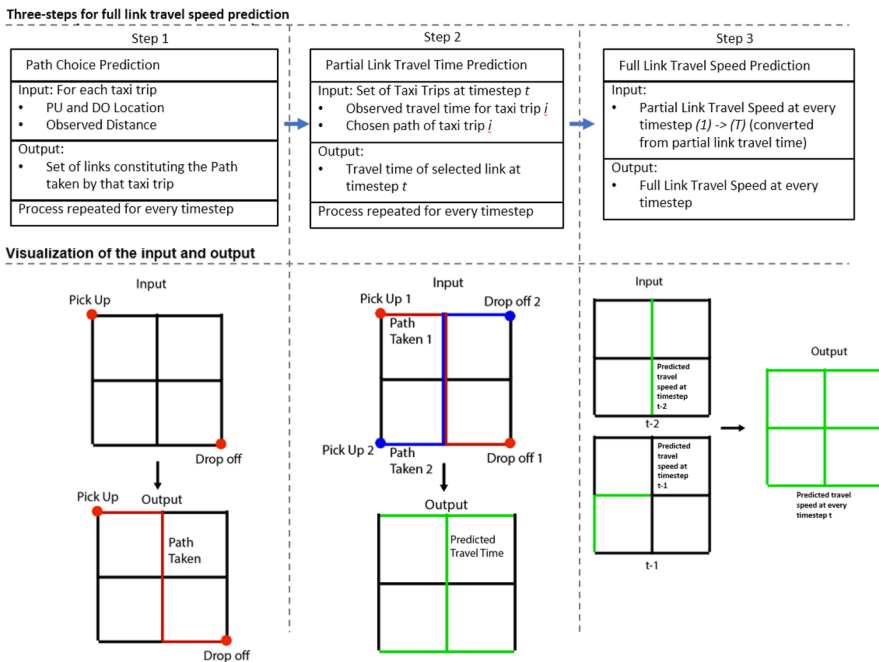


Fig. 1 Schematic Representation of The Sequential Three Steps Framework

from the date 01/01/2014 to 12/31/2015) are executed. Therefore, we do not include the subscript of time window t in the description and formulation of step 1 and 2. The S3S framework is the key component of overcoming the incompleteness of taxi trip data as mentioned in Sect. 2.

3.3 Framework's Application on Similar Data Types

Besides a single taxi dataset with only the pickup and dropoff coordinates available, our research also works well or even better with similar but denser spatio-temporal dataset. One such example is the extension of taxi dataset including GPS traces. GPS traces provide two fundamental advantages over the original taxi dataset which are (1) the path chosen by the taxi between the Origin–Destination pair is known and (2) partial link travel time can be derived directly by comparing the timestamp between the start and end of a road segments for a taxi trip. Therefore, our framework can bypass step 1 and 2 completely because the path choice and partial link travel time derived from GPS traces is theoretically more accurate than the estimation from step 1 and 2. GPS traces can then be introduced directly to the input of step 3. The disadvantages of using GPS traces are (1) such data might not be available in some city due to privacy reason; and (2) the GPS-implemented fleet is biased within certain areas and does not cover the entire network. Our framework provides flexibility and robustness by including step 1 and 2 and practitioner can make a choice based on the data they have.

3.4 Step 1: Path Choice Prediction (PCP)

Given the input of the pickup and drop-off GPS coordinates and the observed travel distance of a taxi trip, this step aims to infer the path taken by that taxi trip. PCP step includes the following procedure. First, the pickup and drop-off coordinates are projected to the nearest link via a perpendicular line. This projected point is called the mapped point. Then the mapped point is projected to the nearest intersection which is then named as an intermediate node. We assume that the taxi would not make a U-turn for either picking up or dropping off customer which results in only one unique pair of intermediate nodes for pickup or drop-off. Figure 2 shows both a schematic view and real-world application of the data mapping step. We also calculate the distances between the map points and intermediate nodes, which are named D_1 and D_2 respectively. Second, for each pair of intermediate nodes, we generate k -shortest-paths using Yen's Algorithm (Yen 1971) and calculate the traveling distance for each path. We need to add this travel distance, D_1 , and D_2 together in order to get the predicted path distance. A pseudocode for Yen's Algorithm is provided in Appendix B. One drawback of Yen's Algorithm is the paths generated only vary slightly between each other and there are a lot of overlap links. In the real world, drivers are often presented with a diverse set of alternative paths with limited overlapping and the Yen's Algorithm cannot easily captures this notion. There are other algorithms aiming at improving speed or increasing diversity such as Hoffman's Algorithm (Hoffman and Pavley 1959), Multi-pass (Chondrogiannis

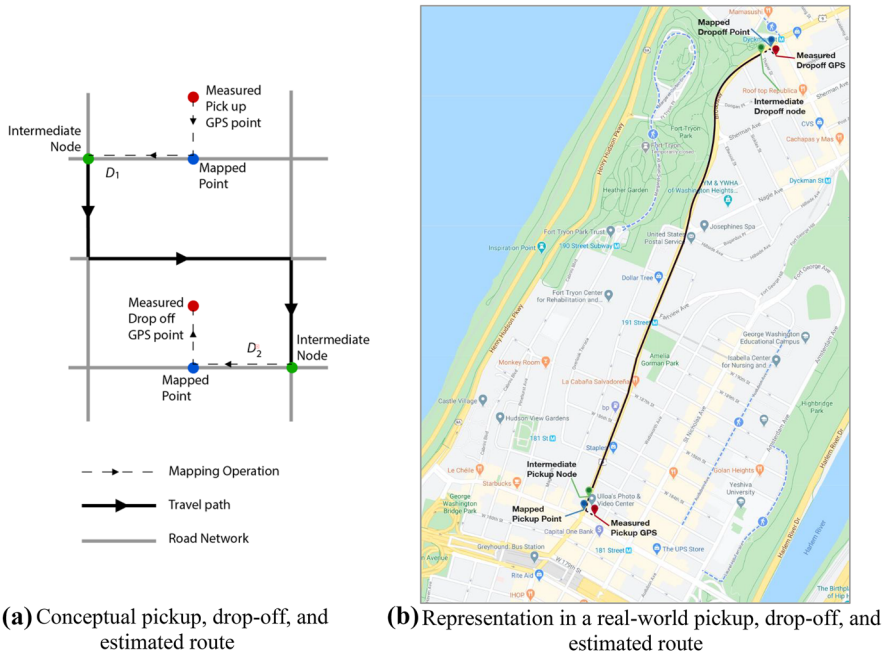


Fig. 2 Data Mapping

et al. 2020), Constrained Time-Dependent KSP Algorithm (Hu and Chiu 2015), or Greedy Framework (Liu et al. 2018). However, these algorithms are complex and only being tested in moderate size network. Our case study features the New York City network with more than 9,500 links and 4,500 nodes, which can pose difficulties for those algorithms. In addition, the k -shortest-path problem need to be solved for every single taxi trip and the computational time and power for such algorithm is simply not applicable. The Yen's Algorithm is the only algorithm that can practically be applied in this situation. To alleviate this drawback, we exclude taxi trip records with too high of an error between the observed and predicted travel distance from the training set. The choice of k is a model's hyperparameter and in our New York case study, after several trials, we choose $k=5$ since it balances between computational time and model's accuracy. Finally, the taxi trip's chosen path is the path minimizing the absolute difference between the predicted path distance and the observed distance collected from the taxi trip record. An example of Path Choice Prediction is also shown in Fig. 2. Step 1 is executed independently and repeated for each taxi trip and each time period.

3.5 Step 2: Partial Link Travel Time Prediction (PLTT)

Given the input taxi trips including its predicted chosen path and observed travel time, this stage aims to infer the travel time of only selected links. Step 2 is also

executed independently and repeated for each time period. In the main formulation of PLTT, the decision variable is the partial link travel time, and the constraint is the predicted trip travel time, which is computed directly from partial link travel time, is close to the observed trip travel time. Therefore, the number of decision variable and constraints are number of links included in the taxi set and number of trips respectively. However, if we apply PLTT directly to a set of taxi trips, the number of decision variables will be much greater than the number of constraints and the resulting partial link travel time is not as accurate. Therefore, we first create a preprocessing Taxi Trip Sub-setting (TTS) model to address this problem. TTS aims to maximize the number of trips selected for subset S while guarantees that the number of selected trips divided by the number of involved links is greater than a certain threshold. The decision whether a taxi trip record i is selected for subset S is represented by the decision variable x_i of the TTS model and the formulation is as follows:

3.6 TTS Model

3.6.1 Objective Function

$$\text{Max} : \eta = \sum_{i \in I} x_i \quad (1)$$

3.6.2 Subject to

$$L = \sum_{a \in A} \left[1 - \prod_{i \in I} (1 - P_i x_i) \right] \quad (2)$$

$$\eta/L \geq \beta \quad (3)$$

Equation (1) is maximizing the number of selected trips for the subset S . Equation (2) shows the total number of involving links L corresponding to the subset S . P_i is a vector of size $A \times 1$ representing the sequence of link constituting the travel path for taxi trip i . This means that if link a is presence in the path sequence of trip i , the a^{th} value of vector P_i is equal to 1 and 0 for otherwise. The decision variable x_i is binary showing whether trip i is selected for the subset S . The product \prod is an element-wise vector multiplication across every trip i of the vector $(1 - P_i x_i)$. The \sum is the sum of all elements in the resulting vector from the previous calculation and it is equal to the number of links included in subset S . Equation (3) ensures that the ratio between the number of trips selected and its corresponding number of participating links is larger than a hyperparameter $\beta \in (0, 1)$. A higher β and thus, a higher ratio of η/L would yield a more accurate result for PLTT because the number of constraints would be close to the number of decision variables. However, higher β would also narrow down the feasible space and ultimately, fewer links would be selected for prediction and step 3 will receive fewer inputs.

After the subset S is created, the second procedure of step 2 is to predict travel time of the link included in subset S based on the Partial Link Travel Time (PLTT) model. The decision variable tt is a vector of size $A \times 1$ where the a^{th} element represents the travel time of link a . The model aims to minimize sum of square error between the predicted travel time and the observed travel time among all taxi trips in subset S .

3.7 PLTT Model

3.7.1 Objective Function

$$\text{Min} \sum_{i \in S} \left(t_i^{pred} + \Delta_i - t_i^{obs} \right)^2 \quad (4)$$

3.7.2 Subject to

$$t_i^{pred} = \sum_{a \in A} tt \cdot P_i \forall i \in S \quad (5)$$

Equation (4) is the objective function minimizing the sum of square error between the predicted travel time and the observed travel time over all taxi trips i belong to subset S . The predicted travel time only accounts for the summation of link travel time along the selected path and thus, we introduce the term Δ_i to represent the total intersection delay for taxi trip i . Since only the full path travel time is known, we currently set this Δ_i to 0 and the intersection delay is incorporated into link travel time. However, in future research where taxi GPS traces are available, this Δ_i can be accurately determined and improve the accuracy of step 2. Equation (5) shows the predicted travel time of taxi trip i is equal to the sum among all links a the element-wise product $tt \cdot P_i$. This entire step is repeated for each time period.

3.8 Step 3: Dynamic Urban Link Travel Speed Estimation

Step 3 takes the input of partial link travel time from step 2 to produce an output of dynamic traffic state which is represented by link level travel speed. Step 3 procedure is as follows:

1. Encoding link speed as node features: this step aims to transfer link feature, which in this case is travel speed, as node features to facilitate message passing between nodes.
2. Creating a modified directional adjacency matrix: this step creates a matrix that represent simultaneously node connectivity and the relation between different directional flows.
3. Traffic Graph Convolution Network (TGCN): this step develops a convolution operation on graph structure to learn the spatial relation between nodes.

4. Recurrent TGCN: this step joins hidden layers at different time period together to learn the temporal relation between nodes. The computational graph of step 3 would be shown after the discussion of each individual component.

3.8.1 Encoding Link Travel Speed as Node Feature

First, we convert the link travel time to travel speed as the input for the deep learning model since it has the following advantages over travel time:

- Link travel speed are highly correlated between road segments. Thus, it makes the process of message passing between node more efficient.
- In case the link is untraversable, the link travel speed can be set to zero and effectively stop the process of message passing between node whereas for link travel time, one must set an arbitrary large number, which can be subjective.

Recent development of graph convolution network only allows for message passing between node whereas link features such as the distinction between directional movement (i.e., north-bound vs south-bound traffic) are usually neglected. In addition, the graph considered tends to be undirected. This paper aims to bridge this gap by embedding link features (travel speed) as node features. For a node i , there is a maximum of 4 outward link going from node i to its neighbor. The speed on these outward links would be embedded to the node features vector as shown in the example in Fig. 3. Here, travel speed on outward links of node i , which are v_1, v_2, v_3 and v_4 , are embedded to node i feature representation vector:

$$X_i = [x_i^1, x_i^2, x_i^3, x_i^4] \text{ where } x_i^f = v_f, \forall f \in (1 : 4)$$

The inward link to node i should not be embedded because it would be repetitive since node i 's neighbors have already embedded it in their feature representation vector.

The order in which these link travel speed goes into the node feature vector depends on its north bearing angle α_a^{link} and this node-link embedding principal is presented in Table 1. The principal along with the modified directional adjacency matrix help our framework preserving the notion of direction which is inherent in traffic flow.

Fig. 3 Visualization of Link Travel Speed Embedding

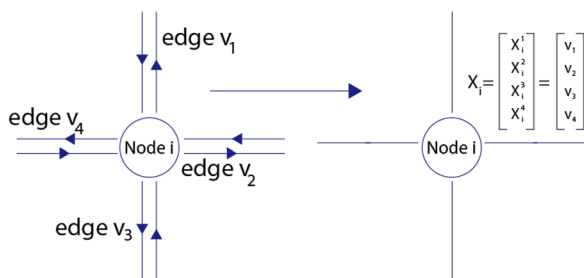


Table 1 Node-Link Embedding Principle

North bearing angle α_a^{link}	Embedding Principle
$0 \leq \alpha_a^{link} < 90$	Link travel speed embedded to the first element
$90 \leq \alpha_a^{link} < 180$	Link travel speed embedded to the second element
$180 \leq \alpha_a^{link} < 270$	Link travel speed embedded to the third element
$270 \leq \alpha_a^{link} < 360$	Link travel speed embedded to the fourth element

3.8.2 Modified Directional Adjacency Matrix

In the normal Graph Convolution Network by Kipf and Welling (2017), the adjacency matrix can only represent the node connectivity and the graph is undirected. This is not applicable for road network because the direction at which these nodes are connected is important. This can be shown by the following example:

We consider node 2 in Fig. 4a as the node of interest or host and it is looking to gain information from its neighbor, node 1. We can say that the value of x_1^2 is highly related to x_2^2 while none other pair is related. Therefore, only the pair of x_i^f that has $f=2$ can pass message between each other. This is because the direction of the link from neighbor 1 to host 2 lies in the $f=2$ direction. This conforms with the node-link embedding principle mentioned in Table 1. Therefore, we construct a $[N \times N \times F]$ modified adjacency matrix A as follows:

$$A : a_{ijf} = \begin{cases} 1 & \text{if the link from neighbor } j \text{ to host } i \text{ is in the } f \text{ direction} \\ 0 & \text{for otherwise} \end{cases}$$

The adjacency matrix for the example can be visualized in Fig. 4b. Here, every links in the network is embedded separately even for those with the same OD pair and $a_{ijf} \neq a_{jif}$. The notion of direction is preserved with the use of the third-dimension f . The adjacency matrix, node-link embedding principle, and the traffic graph convolution network are specifically designed to be compatible with each other and capture the real-world network traffic as close as possible. In the GCN introduced by Kipf & Welling (2017), the adjacency matrix is normalized using the following operation to ensure the node features after convolution are not scaled up:

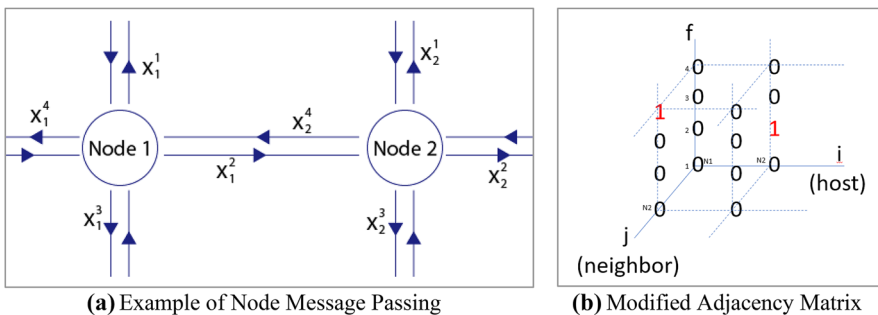


Fig. 4 Node Message Passing

$$\bar{A} = \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}$$

where: $\hat{A} = A + I$, and I is an identity matrix and \hat{D} is diagonal node degree matrix of

This normalization ensures that the row sum of \bar{A} is equal to 1. However, this normalization is not necessary for our Modified Directional Adjacency Matrix. Within a certain direction f , we only want the host node i to look at one particular relevant neighboring node j and thus, only this pair a_{ijf} value is 1 whereas the remaining is 0. Therefore, this guarantee:

$$\sum_j a_{ijf} = 1, \forall i \in I, f \in F$$

To help explain this feature, we refer to back to Fig. 4 and assume the host node is node 2 and the direction of interest is $f=2$ (or East). Here, we only allow x_1^2 of node 1 to pass information to node 2's x_2^2 . Suppose there are other nodes neighboring node 2 to the East, North, and South. However, none of these neighboring nodes can pass information about x_2^2 and thus, their a_{ijf} value is 0. Therefore, for a particular direction, this guarantees the row summing to 1 for our Modified Adjacency Directional Matrix.

3.8.3 Traffic Graph Convolution Network (TGCN)

After embedding link to node and constructing the directional adjacency matrix, TGCN will apply “convolution” over the road network. GCN can be described as a neural network where each node of the graph is a neuron itself and the propagation rule is the “convolution” operation. In CNN, “convolution” refers to aggregating information from nearby pixels. Therefore, the idea of “convolution” on a node can be regard as aggregating the information from all of its neighbor. This can be visualized in Fig. 5a where the host node u is looking to gain information from its immediate neighbor v_1, v_2, v_3 , and v_4 .

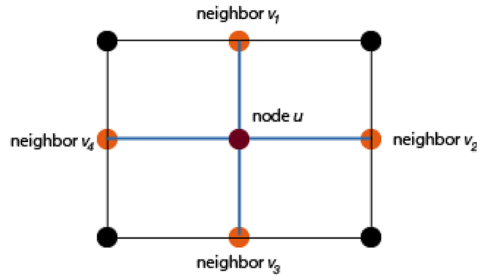
Therefore, the Graph Convolution on graph domain can be formulated as follows:

$$h_{u,f}^k = \text{relu} \left(\sum_{v \in N(u)} w_{uv} h_{v,f}^{k-1} + b_u \right), \forall u \in N, f \in F \tag{6}$$

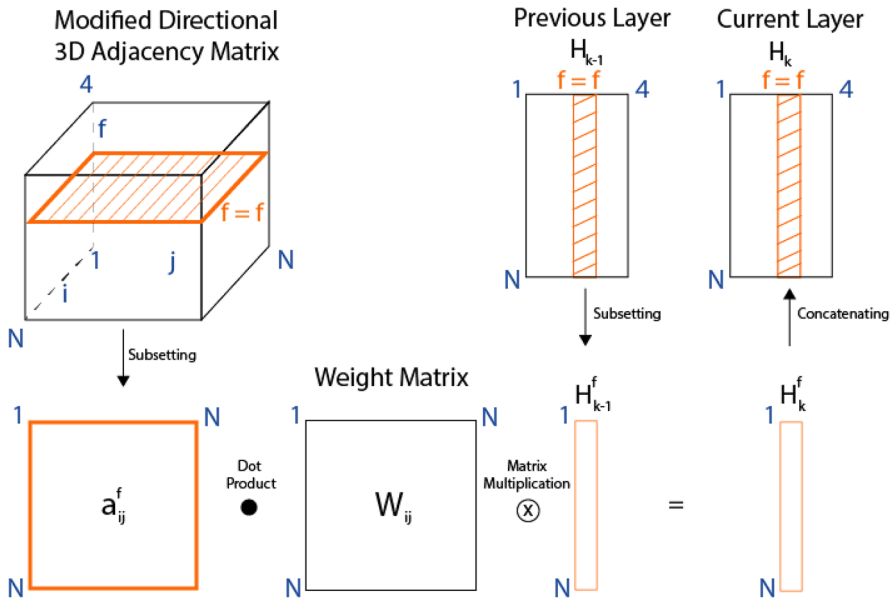
where $\text{relu}(x)$ is the rectified linear activation function (ReLU) as follows:

$$\text{relu}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Equation (6) represents the convolution on graph where for every neighbor v of u , we take the previous hidden layer of neighbor $h_{v,f}^{k-1}$, multiply it with the weight w_{uv} , and aggregate it to the current layer of host u . In addition, a bias b_u is added and the entire operation is transformed using the ReLU function. Here we do not need to add the subscript f for the weight w_{uv} because for a pair of host u and neighbor v , there is



(a) Convolution on Graph Domain



(b) Graph Convolution on Matrix Domain

Fig. 5 Convolution on Graph

only one link in a specific f direction (e.g., $f=2$ in the example of Fig. 4) that can facilitate node message passing. This operation is repeated for each node u within the network and for each traffic directional flow f . This repetition can pose a problem in computational efficiency and Eq. (6) is very difficult to be programmed. Therefore, we modify and migrate the operation from graph domain to matrix domain. First, instead of using the individual node representation $h_{u,f}^k$, we use a $[N \times F]$ matrix H_k where each row represents the node feature. In similar fashion, a $[N \times N]$ matrix W and a $[N \times F]$ matrix B would be utilized instead of the set of individual weights w_{uv} and b_u . The modified directional adjacency matrix A would facilitate the transition from graph domain to matrix domain. Equation (6) can be rewritten in matrix domain shown in Eq. (7) below.

$$H_{k,pre_transformed}^f = a^f \odot W \otimes H_{k-1}^f, \forall f \in (1 : 4) \tag{7}$$

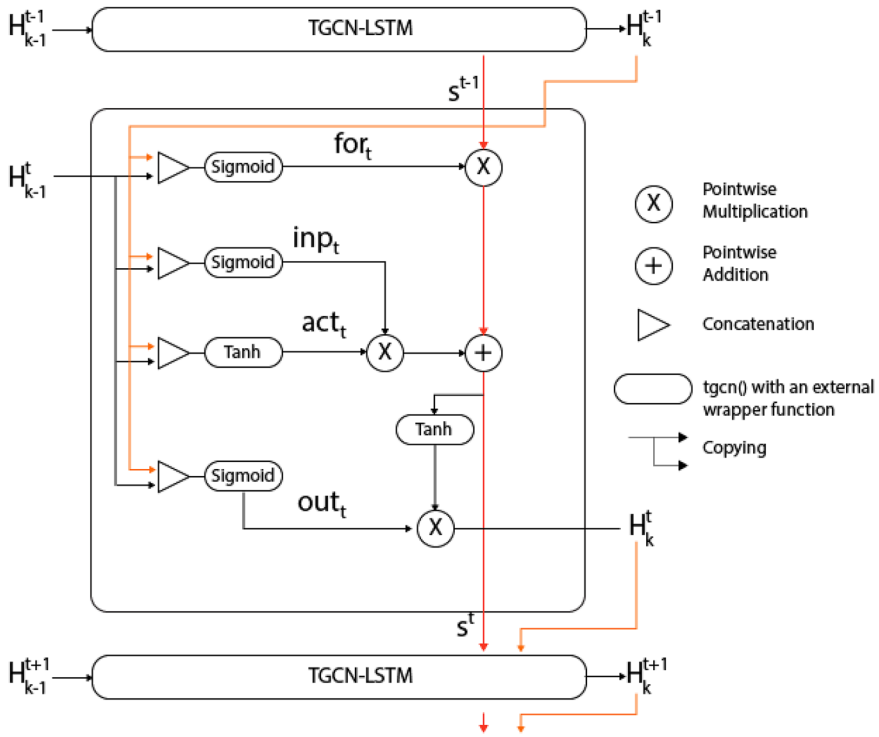
$$H_k = \text{relu}(H_{k,pre_transformed} + B) \tag{8}$$

Here, for a specific value f , we first subset a matrix a^f from the modified adjacency matrix A and perform an element-wise matrix multiplication with W . The resulting matrix is then used as the first component in the matrix multiplication with the vector H_{k-1}^f subsetted from the previous layer network state H_{k-1} at f . A visualization of Eq. (7) is shown in Fig. 5b. This operation is repeated for every $f \in (1 : 4)$. Equation (8) adds the bias B and the entire operation is transformed using the ReLU function and the resulting matrix is the current layer network state H_k . By performing Eqs. (7) and (8), a node will effectively learn the information from its neighbor one hop away. TGCN can be repeated multiple times to allow the node learning from neighbors further away. Equations (7) and (8) address the research gaps in Sect. 2 and it has the following three advantages: (1) It preserves the notion of directional flow, (2) It transform a complex convolution on graph domain to matrix domain and make it readily for scaling to larger network, and (3) the weights W can be shared across multiple execution of TGCN and thus greatly reduce the number of weights to be trained. This weight sharing feature will be discussed further in the computational graph section below.

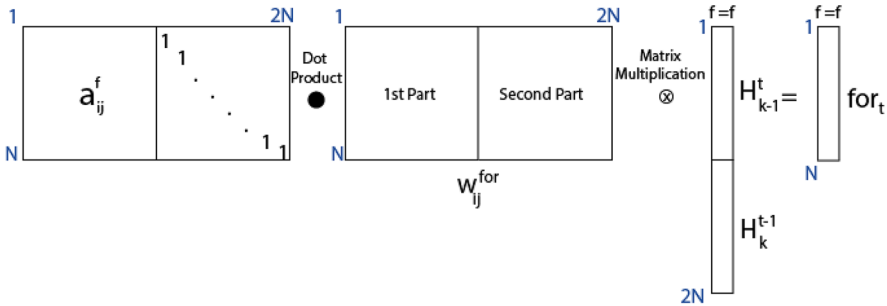
3.8.4 TGCN_{lstm} for capturing Temporal Dependencies

After the TGCN operation is applied at each time period to capture spatial relation, we introduce the TGCN_{lstm} cell to join the traffic states at different time periods together. We modify the well-known LSTM cell to a TGCN_{lstm} cell to capture both temporal and spatial relation in one operation and apply it to the final two layers $K - 1$ and K . The TGCN_{lstm} cell would take three inputs which are (1) the current layer at previous time period H_k^{t-1} , (2) previous layer at current time period H_{k-1}^t , and (3) the internal state at previous time period s^{t-1} . There are two outputs of the TGCN_{lstm} cell namely (1) the current layer at current time period H_k^t and the internal state at current time period s^t . The computation graph of LSTM can be shown in Fig. 6a as follows:

In the spatial aspect of TGCN_{lstm}, it functions in similar fashion as TGCN with the only difference being TGCN_{lstm} has two inputs H_k^{t-1} and H_{k-1}^t . Therefore, we would have to modify the adjacency matrix a_{lstm}^f , the weight matrix W_{lstm} , and the convolution operation as shown in Fig. 6b. First, the previous layer at current time period H_{k-1}^t and the current layer at previous time period H_k^{t-1} are row-wise concatenated resulting in a $[2N \times F]$ matrix. We column-wise concatenate the adjacency matrix a^f at $f = f$ with an $[N \times N]$ identity matrix to get the adjacency matrix a_{lstm}^f for TGCN_{lstm}. The weight matrix is also in $[2N \times N]$ dimension where the first $[N \times N]$ part is the same as TGCN and the second $[N \times N]$ part only has weights in the diagonal location and the remaining location are 0. We then perform the dot product between a_{lstm}^f and W_{lstm} . The resulting matrix is the first component of the



(a) TGCN_{lstm} Cell Architecture



(b) $tgcn_{lstm}()$ operation

Fig. 6 TGCN_{lstm} Architecture

matrix multiplication, and the second component is the subset at $f = f$ of the concatenated input $[H_{k-1}^t; H_k^{t-1}]$. The process is repeated for every f . A schematic representation of which is shown in Fig. 6b. Despite the concatenation, the resulting matrix will have N rows thus make it easily compatible with other operation. This adjustment encourages the node to not only takes temporal information from itself but also from its neighbor. We name the entire operation as $tgcn_{lstm}()$ as follows

to better describe the calculation of the internal values occurs within the $TGCN_{lstm}$ cell.

$$H_{k-1}^{t,f} = tgcN_{lstm}\left(W_{lstm}, \left[H_{k-1}^{t,f}, H_k^{t-1,f}\right]\right) = a_{lstm}^f \odot W_{lstm} \otimes \left[H_{k-1}^{t,f}, H_k^{t-1,f}\right], \forall f \in (1 : 4)$$

From the computational graph shown in Fig. 6, we have 4 internal parameters namely for_t , inp_t , act_t , and out_t and these parameters are calculated by the following equations:

$$for_t = \sigma\left[tgcN_{lstm}(W^{for}, [H_{k-1}^t, H_k^{t-1}]) + B^{for}\right] \tag{9}$$

$$inp_t = \sigma\left[tgcN_{lstm}(W^{inp}, [H_{k-1}^t, H_k^{t-1}]) + B^{inp}\right] \tag{10}$$

$$act_t = \tanh\left[tgcN_{lstm}(W^{act}, [H_{k-1}^t, H_k^{t-1}]) + B^{act}\right] \tag{11}$$

$$out_t = \sigma\left[tgcN_{lstm}(W^{out}, [H_{k-1}^t, H_k^{t-1}]) + B^{out}\right] \tag{12}$$

$$s_t = for_t \cdot s_{t-1} + inp_t \cdot act_t \tag{13}$$

$$H_k^t = out_t \cdot \tanh(s_t) \tag{14}$$

where $\sigma()$ is a sigmoid function and $\tanh()$ is a hyperbolic tangent function as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Equations (9–12) represent the calculation of the internal parameters for_t , inp_t , act_t , and out_t respectively. In each of these equations, a new set of trainable weight and bias are introduced. These weights and biases are shared across time periods. Equation (13) determines the internal state value at current time period s_t . This ensures the gradient flow is uninterrupted across time periods. Equation (14) shows the final network link travel speed at current time period H_k^t .

3.8.5 Computational Graph

In this section, we present the computational graph stating the flow of information from the input to the output. We choose the number of layers as $k=[0:4]$ and $t=[1:28]$ for our New York City Case study. A training sample would represent a typical day (i.e., Monday 01/06/2014). First, the input of a typical sample is disaggregated into multiple states H_0^t where each state represents a specific

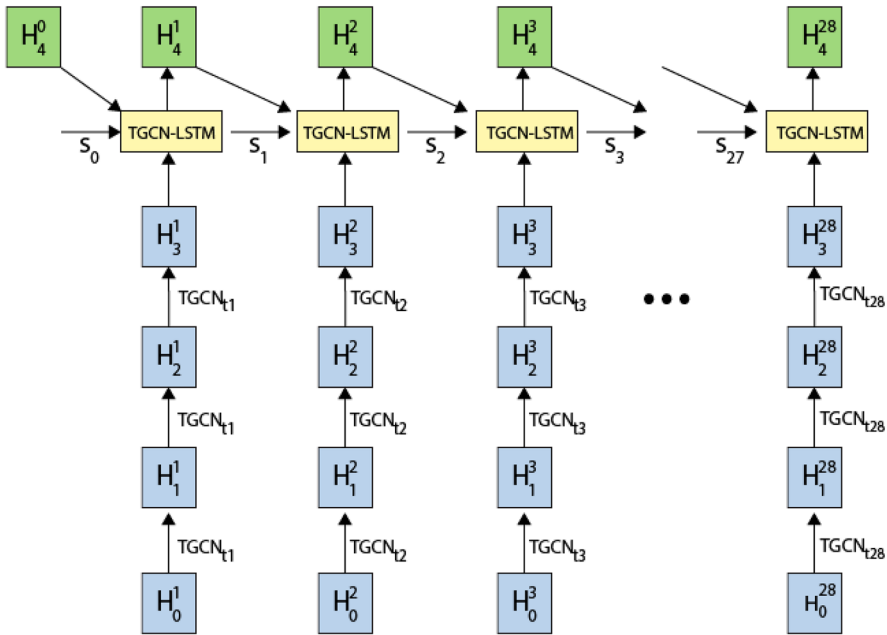


Fig. 7 Computational Graph

time period (i.e., 06:00 – 06:30). In the example of our New York City network, our analysis period ranges from 06:00 in the morning to 20:00 in the evening at 30 min interval which results in 28 states of H'_0 . A combination of these 28 H'_0 states will form one training sample. The input would be partial link travel speed from step 2. If the speed of the link is not available, the value is set to 0 thus disrupts the message passing from the beginning. Then, the TGCN are applied three time to H'_0 at each time period resulting in H'_3 . The TGCN’s weight W is shared from H'_0 to H'_3 but not across different time period. This means that the TGCN’s weight W is distinctive for each time period. Therefore, we differentiate it by the subscript $TGCN_t$. At the last two layers, the $TGCN_{lstm}$ takes the input of H'_3 , H'_4 , and s^{t-1} and produces the outputs of H'_4 and s^t . Here, the set of weights and biases for $TGCN_{lstm}$ are shared across time periods. The final layer H'_4 corresponds to the final output of network wide dynamic link travel speed. In this computational graph, a node can gather information from neighbors as far as 5 “hops” away. Figure 7 shows the full computational graph of the DU-LSE model.

The loss function is a sum of square error (MSE) between the predicted travel speed and the observed travel speed, and it is accumulated over all H'_4 . The loss is computed on the observed values only and it is computed in Eq. 15 shown below:

$$Loss = \sum_{t \in T} \left(H_4^{t,obs} - H_4^{t,pred} \right)^2 \tag{15}$$

4 New York City Case Study

4.1 New York Taxi Dataset

Our study's methodology can benefit cities possessing taxi trip data such as Chicago or San Francisco and we choose New York City to demonstrate our framework. The Taxi Trip dataset used in this paper is obtained from the New York City Taxi and Limousine Commission (NYC-TLC) and it is publicly available on the City of New York's official website.¹ Each observation of the dataset is a single taxi trips which includes several details of the trip and our paper mainly focuses on the following four: (1) Pickup and Dropoff Coordinates; (2) Pickup and Dropoff Timestamp; (3) Observed Travel Distance; and (4) Observed Travel Time. The coordinates are only available from 2015 backwards and thus we select January 1st, 2014, to December 31st, 2015 as our analysis period. We load the taxi trips dataset into a local database using PostgreSQL to facilitate the task of querying specific records falling into a time period. Our daytime analysis period ranges from 06:00 – 20:00 with a 30-min interval resulting in 28 time period. Upon doing descriptive statistics, we recognize the difference in travel pattern between the weekdays is significant enough that we would need to create several models for each set of day. This is shown in Fig. 8b below. Due to limited resources, we only consider Monday of each week for analysis, but the methodology can be readily applied to other weekdays as well. We select the entire Manhattan road network for analysis which has 9,617 road segments or links and 4,750 nodes or intersection. The network is obtained from the OpenStreetMap Database.² Therefore, we select taxi trips of which both pickup and drop-off coordinates fall within the Manhattan network. In addition, we present several visualizations of the dataset from both a spatial and temporal perspective. Figure 8a shows the spatial aspect of the dataset where the Manhattan Network and the Taxi Trip Pickup and Dropoff location are presented. On the other hand, Fig. 8b shows the temporal aspects where the average number of trips and travel speeds are presented.

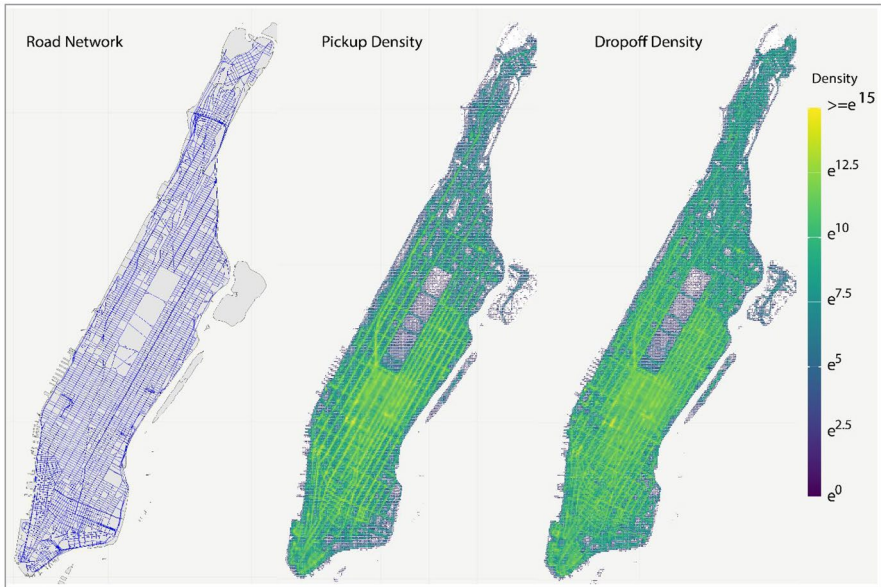
4.2 Results

First, we choose the following values for the hyperparameters in the New York Taxi case study. We set $k = 5$ in the k^{th} shortest path algorithm; ratio threshold $\beta = 0.85$ in the TTS model; and intersection delay $\Delta_i = 0$ in the PLTT model of step 2.

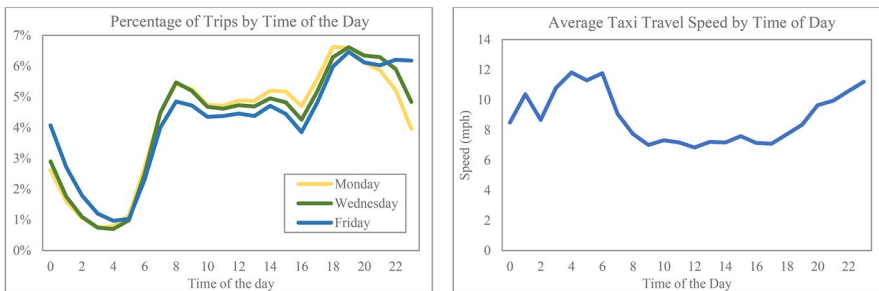
To implement the methodology discussed in the previous section, we use multiple means of programming. First, the raw data in.csv format is stored in a PostgreSQL database for easy querying. For the first two steps of Path Choice Prediction and Partial Link Travel Time Prediction, we use R programming with built in library for Parallel Computing and Optimization Algorithm. For calculating k^{th}

¹ <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

² <https://www.openstreetmap.org/relation/175905>



(a) Spatial visualization of the dataset including: road network, pickup density, and dropoff density



(b) Temporal visualization of the dataset

Fig. 8 Descriptive Statistics of the Dataset

shortest path, we use a function from PostgreSQL which is `pgrouting` and it uses Yen algorithm for this computation. The first step is computationally heavy since the computer needs to do Data-mapping and finding k-shortest-path for every taxi trip records. However, the process is accumulative and the “divide and conquer” strategy is applicable. After finishing the first two steps, we use Python programming, specifically the library TensorFlow 2.0 (Google 2020), to develop our deep learning model.

In Step 3, the model takes the input of Partial Link Travel Time from Step 2, converts it into input speed, and produces the output of network wide link travel speed. In our New York city network case study, the size of the network is 9,617 link and the number of time period is 2,912 (note that 28 time period would comprise a day). The output speed is available across all these links and time periods. However,

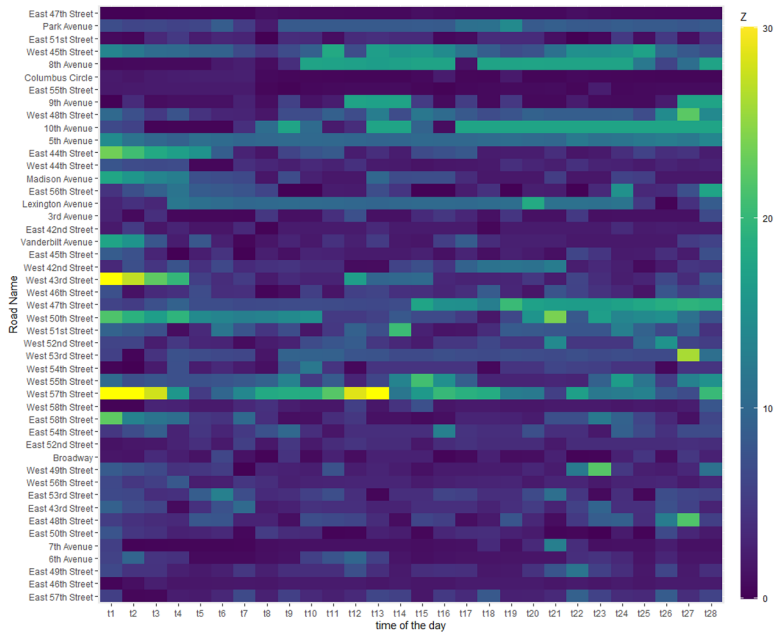
it would be difficult to show all these values and we decide to illustrate our result for selected links and time periods only. We choose the time periods on Monday, 06/23/2014 to show our result and this date is chosen randomly. Figure 9 shows the heatmap of step 3 input and output. Each row represents a specific road segment, and each column represents each time period from 06:00—20:00 at 30 min increment. The road segments are taken from the Midtown Area just below the Central Park. The heatmap color scale is provided on the right representing travel speed in miles per hour.

The color in each cell represents the link travel speed. For the input heatmap in Fig. 9a, links, of which travel speed is not available, are colored in white. Cells tilting toward dark purple denote a lower link travel speed and those tilting toward brighter yellow denote a higher travel speed. From Fig. 9b, the majority of links experience travel speed less than 15 mph in most time of the day and only a few links such as West 57th Street or West 43rd Street have travel time larger than 25 mph during the earlier period of the day (i.e., 06:00—07:30). For each link, we also see a gradual change in travel speed throughout time of the day where most link start at a higher travel speed, decrease and remain low during working hours, and increase back in later in the evening.

Figure 10 shows the temporal aspect of our model's result where link travel speed at each time period is illustrated. To demonstrate the spatial aspect, we plot two subsets of the New York city network representing major arterial and minor arterial in the Midtown area respectively in Fig. 10. For each subset, three time period namely 07:00—07:30, 12:00—12:30, 18:00—18:30 are selected since these time periods represent different trip purpose (i.e., commuting, lunch, and recreation) and vary greatly in the Origin–Destination trip demand. In the computational graph shown in Fig. 7, the TGCN module's training weights vary between time period which gives the model flexibility in capturing different network speed patterns emerging throughout time of the day and this flexibility is reflected in Fig. 10. The link travel speeds are color-coded with the same analogy as in Fig. 9 and the color scale is provided on the right. In the subset for major arterial plots, the northern part of Manhattan has considerably higher travel speed than the rest of the network whereas the Midtown and Downtown area experience consistent low travel speed with some exceptions such as the highways near the Brooklyn Bridge. In overall network performance, the 07:00—07:30 time period rank first in travel speed, followed by the 18:00—18:30 and the 12:00—12:30 time period as illustrated by the color tilting from the "brighter" side to the "darker" side. Figure 10b shows the subset of minor arterial in the Midtown area accompanied by its location with respect to the Manhattan Network as a whole. The majority of the link has travel speed less than 15 mph through all three time period. One small observation is the two long links in the bottom right of the network which has relatively high travel speed as compared to the remaining links. These two links are part of the Park Avenue Road at the segment near Grand Central Terminal and these links are on a different elevation compared to the rest. The curve link in the middle of the network is Broadway where the southern segment is utilized as walking avenue which explains the sudden stop in this link. The first northern segment before West 57th street of Broadway Avenue has higher travel speed compared to the rest of the segment. After West 57th and downward, there



(a) Speed Input of Selected Links from 06:00 – 20:00



(b) Speed Output of Selected Links from 06:00 – 20:00

Fig. 9 Heatmap Showing Input and Output Speed of Selected Links

road geometry is expanded from a normal two-lane one direction road into multiple purpose road with two side parking and bike lane available. These segments also have a higher density of retail areas, restaurants, and offices too.

5 Model Evaluation and Validation

The first part of this section evaluates the model's performance internally by comparing the prediction result from each step of the framework with the taxi trip data. The second part of this section validates the entire framework's result with the ground truth data.

5.1 Model Evaluation

In this section, we discuss the evaluation for every three step of the model's sequential framework. Step 1 aims to predict travel path by minimizing absolute difference between the predicted distance from the travel path generated from K-Shortest-Path algorithm and the observed travel path recorded by the taxi. Figure 11a shows the distribution of the absolute error between predicted and observed value of step 1 at three different time periods. We disaggregate the plot by trip distance because for longer trip, we expect a larger error compared to shorter one. The unit of y-axis is density and because the figure is a distribution plot, the total area under the curve sums to 1. In the trip distance 0–4 and 4–8

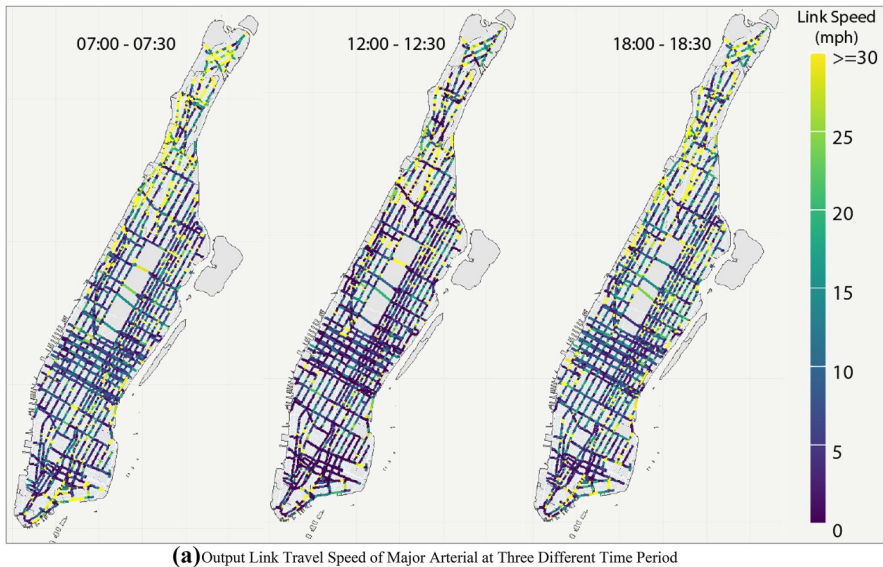
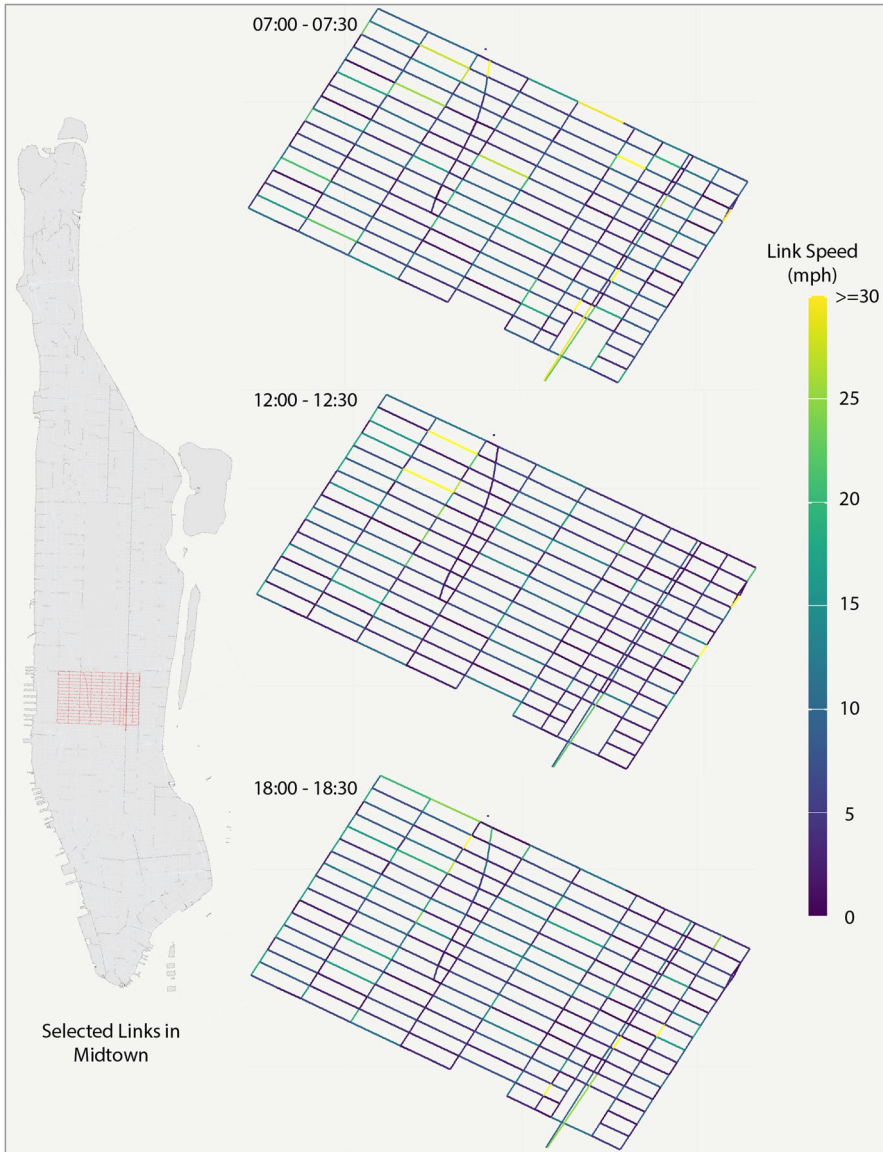


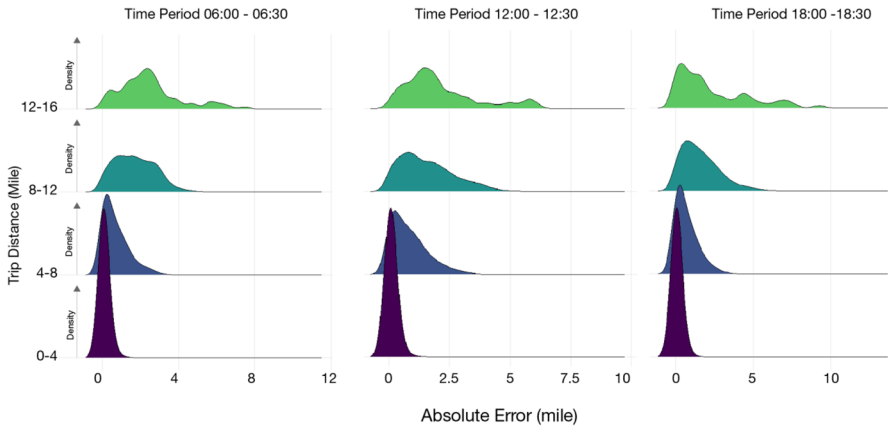
Fig. 10 Output Link Speed of Major and Minor Links at Three Different Time Period



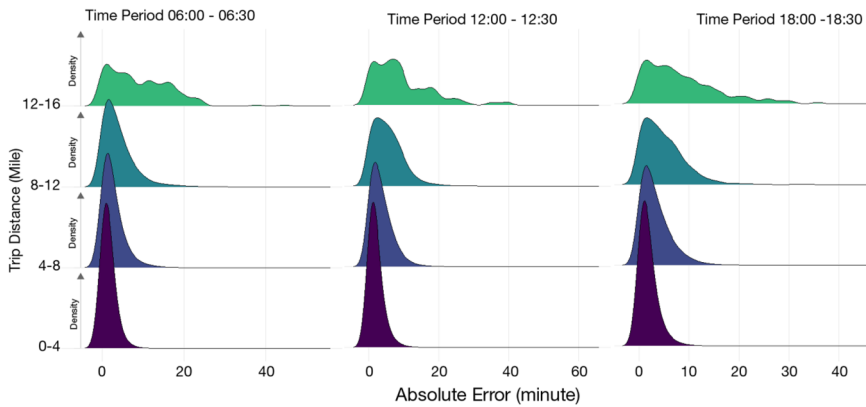
(b) Output Link Travel Speed of Selected Links in Midtown at Three Time Period

Fig. 10 (continued)

category, the distribution is highly centered around the mean value being near zero with a slight skew to the right. In the 8–12 and 12–16 category, the distribution has a longer tail on the right, which indicates some outliers with high absolute error. However, the mean remains at a low enough value. In general, the time period 12:00–12:30 has a better performance compared to the other two. In



(a) Absolute Error between predicted and observed trip distance



(b) Absolute Error between predicted and observed trip travel time

Fig. 11 Absolute Error of Step 1 and Step 2 at Selected Time Period

similar fashion, step 2 aims to predict link travel time at selected links by minimizing the error between observed and predicted trip travel time and the absolute error distribution of which is shown Fig. 11b. We also see the same distribution as in Fig. 11a for the first four trip distance category. Step 2 only selects a fraction of the observation for analysis and by keeping the ratio between number of trips selected and number of links in subset S high, the optimization model is able to achieve high accuracy.

We introduce two metrics namely Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) to evaluate not only this step but also the other two remaining steps and the calculation of these two metrics are shown in Eqs. (16) and (17).

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1:n} (X_i^{pred} - X_i^{obs})^2} \quad (16)$$

$$MAPE = \frac{1}{n} \sum_{i=1:n} \left| \frac{X_i^{pred} - X_i^{obs}}{X_i^{obs}} \right| \times 100\% \quad (17)$$

We then subset each taxi trip i by its Time Period and Travel Distance and calculate these subset's $RMSE$ and $MAPE$ instead of the entire dataset in order to identify any pattern of these two metrics across Time Period and Travel Distance. The resulting evaluation is shown in Table 2 where the rows represent time period, and the columns represent travel distance. As trip distance increases, the value of both $RMSE$ and $MAPE$ increase since the origin–destination are too far apart that the K-Shortest-Path cannot cover all available path. At 0–4 mile of distance, the $RMSE$ and $MAPE$ are around 0.19 mile and 7.5% respectively whereas at 12–16 mile of distance, the values are 3 mile and 19% respectively. In contrast to travel distance, the $RMSE$ and $MAPE$ do not greatly fluctuate which is reasonable since the formulation of step 1 does not involve time period.

We perform the evaluation of step 2 Partial Link Travel Time. In this step, we predict travel time of selected links by minimizing the sum of square of the error between the predicted travel time and the observed travel time of selected taxi trips in subset S and thus, this error is the metric to evaluate the performance of step 2. Unlike step 1, step 2's evaluation focuses more on spatial pattern. Since a taxi trip span across multiple spatial regions and the error is calculated on a trip basis, the total trip error is distributed evenly among the spatial regions. Figure 12 shows the Mean Absolute Error (MAE) and the $MAPE$ at each region. The takeaway of these figures is that we can estimate a taxi trip's error by summing the MAE value of all the regions that it passes by. This shows how certain areas might have lower accuracy in partial link travel time prediction.

On average across all taxi trips, step 2 reaches a performance of 2.6 min in MAE and 18% in $MAPE$. The spatial pattern of MAE is very similar to the $RMSE$'s where higher error is located in the Midtown neighborhood near the Central Park and lower error is located at the Northern part of Manhattan such as Harlem, Hamilton, and Washington Heights neighborhood. This is because of two reasons. First, Midtown neighborhoods have considerably higher not only the number of taxi trips, as shown in Fig. 8a, but also the number of taxi trips passing through this area and thus, there are more variations in the error. Second, in step 2 of PLTT's formulation, we model intersection delay, but the value is assumed to be 0 due to lack of data. The variation in travel time at each intersection between red, green, or even left turn waiting/signal and the accumulation of such variations throughout the trip results in an overestimation of the link travel time. Midtown neighborhoods have shorter road segments and significantly higher number of intersections compare to Northern Manhattan's which results in accumulation of intersection delay and ultimately higher error. However, the error value is still

Table 2 Evaluation of Step 1 Path Choice Prediction

Time Period	0-4 mile		4-8 mile		8-12 mile		12-16 mile	
	RMSE (mile)	MAPE (%)	RMSE (mile)	MAPE (%)	RMSE (mile)	MAPE (%)	RMSE (mile)	MAPE (%)
6:00-6:30	0.194	7.5	0.928	11.7	1.766	15.9	3.046	18.4
6:30-7:00	0.198	7.6	0.929	11.9	1.913	17.9	3.031	19.3
7:00-7:30	0.198	7.4	0.968	12.3	2.009	19.3	2.977	18.7
7:30-8:00	0.197	7.5	1.011	13.1	1.982	19.0	2.797	15.4
8:00-8:30	0.195	7.6	1.039	13.7	2.012	19.3	2.954	19.0
8:30-9:00	0.192	7.6	1.038	13.9	1.940	18.2	3.342	20.8
9:00-9:30	0.192	7.7	1.051	14.2	1.985	18.6	3.036	18.2
9:30-10:00	0.193	7.7	1.070	14.4	1.961	18.3	3.542	23.0
10:00-10:30	0.192	7.6	1.079	14.5	1.984	18.5	3.347	22.0
10:30-11:00	0.191	7.6	1.093	14.7	1.947	18.0	2.757	17.2
11:00-11:30	0.192	7.6	1.101	14.7	1.937	17.9	2.842	17.9
11:30-12:00	0.192	7.7	1.107	14.7	1.891	17.4	2.731	16.6
12:00-12:30	0.191	7.8	1.107	14.7	1.901	17.3	2.631	16.4
12:30-13:00	0.189	7.7	1.085	14.5	1.924	17.7	2.955	18.9
13:00-13:30	0.191	7.7	1.111	14.7	1.890	17.3	2.508	15.1
13:30-14:00	0.189	7.7	1.102	14.6	1.948	17.9	2.506	16.6
14:00-14:30	0.190	7.7	1.093	14.4	1.919	17.5	2.819	17.2
14:30-15:00	0.189	7.7	1.089	14.4	1.933	17.7	3.150	18.0
15:00-15:30	0.189	7.6	1.074	14.1	1.909	17.4	2.577	15.9
15:30-16:00	0.189	7.6	1.052	13.9	1.921	17.6	2.560	14.3
16:00-16:30	0.188	7.5	1.035	13.6	1.891	17.3	2.735	17.1
16:30-17:00	0.187	7.3	1.014	13.4	1.893	17.5	3.166	19.3
17:00-17:30	0.188	7.3	1.001	13.1	1.826	16.6	2.523	15.6

Table 2 (continued)

Time Period	0–4 mile		4–8 mile		8–12 mile		12–16 mile	
	RMSE (mile)	MAPE (%)	RMSE (mile)	MAPE (%)	RMSE (mile)	MAPE (%)	RMSE (mile)	MAPE (%)
17:30–18:00	0.187	7.3	0.988	12.9	1.819	16.5	2.456	14.4
18:00–18:30	0.186	7.2	0.958	12.4	1.849	16.9	2.606	14.6
18:30–19:00	0.185	7.1	0.953	12.3	1.783	16.0	2.095	12.4
19:00–19:30	0.184	7.0	0.946	12.2	1.726	15.4	2.502	14.8
19:30–20:00	0.185	7.0	0.934	11.9	1.698	15.0	2.103	12.0

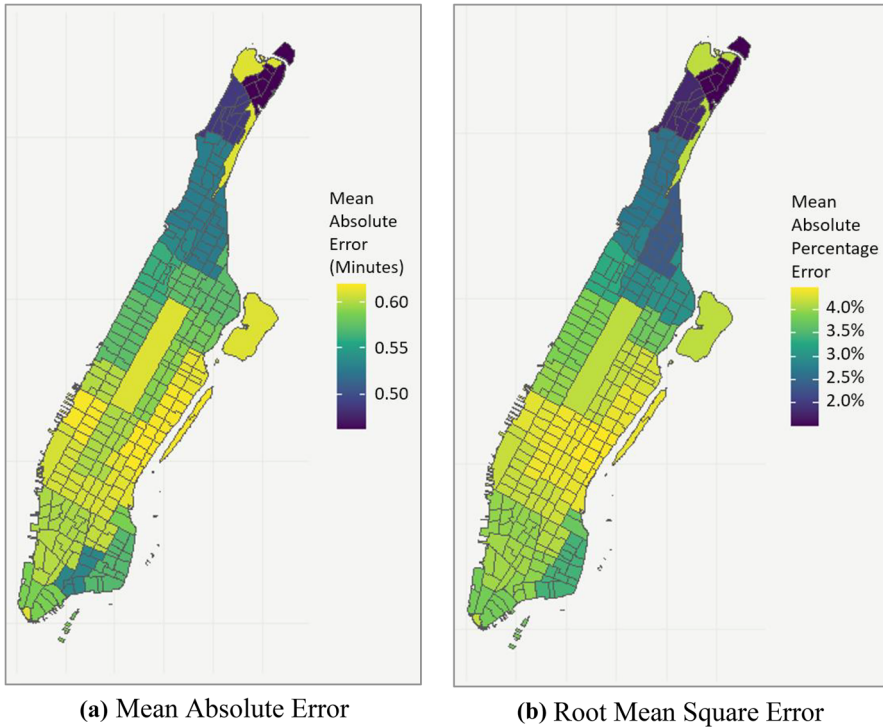


Fig. 12 Step 2's Mean Absolute Percentage Error by Spatial Regions

low at 0.6 min on MAE and 4% MAPE in these areas and it is not significantly more than the minimum error.

To evaluate the Step 3 of Dynamic Urban Link Travel Speed Estimation, we calculate RMSE between the output link travel speed and the partial input travel speed for the observed record of the input only. The RMSE is disaggregated by Manhattan neighborhood and time of the day and the values are shown in Fig. 13. Here, we see a difference in RMSE pattern throughout time period. For time period from 06:00—10:00 the average of RMSE across neighborhoods is 1.8 mph. However, this value decreases considerably later in the day with an average of 0.7 mph at time period 10:00—15:00 and 0.2 at time period 15:00 – 22:00. This is mainly due to the number of taxi trips are much higher from late morning until evening compared to early morning as shown in Fig. 8b. A higher number of taxi trip results in more observed data and better estimation for step 3. The top 3 neighborhoods with the highest accuracy are East Harlem South, Upper East Side, and Gramercy of which average RMSE value are 0.53, 0.60, and 0.61. This is reasonable because it's in the middle of the network and a lot of taxi trip would travel through these areas.

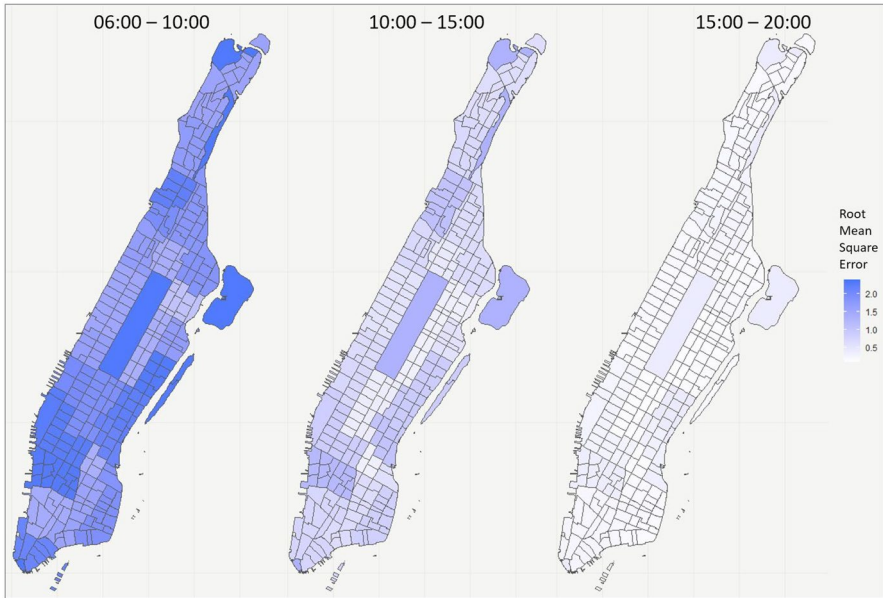


Fig. 13 Evaluation of Step 3: RMSE by Neighborhood and Time Period

5.2 Model Validation

In this section, we first validate our framework with a baseline framework from Zhan et al. (2013). Then, we obtain ground-truth travel speed data collected from traffic sensor by the New York Department of Transportation's Traffic Management Center and compare it with our framework.

5.2.1 Validation with Similar Methodology

We choose Zhan et al. (2013) as the baseline framework for comparison because their paper has the similar scope to our's, which is estimating link travel time using taxi dataset with only the origin and destination known (GPS traces are not available). Their methodology considered path taken by taxi trips as latent and proposed a multinomial logit model to estimate the probability of choosing a particular path. The decision variable is the link travel time, and the expected path travel time are based on this. By multiplying the path probability with its expected path travel time, the paper arrived at the expected trip travel time and the objective is to minimize the root mean square error between the expected and observed trip travel time. Since their methodology is applied to a small road network near Central Park, we also apply their methodology to our taxi dataset on a small network that is identical to the Midtown network in Fig. 10b. From the 2-year taxi dataset, we select 2 days and 3 time periods from each day resulting in 6 instances. The 2 days are April 7 and 20 of 2015 and the 3 time periods are 07:00 – 07:30, 12:00 – 12:30, and 18:00 – 18:30.

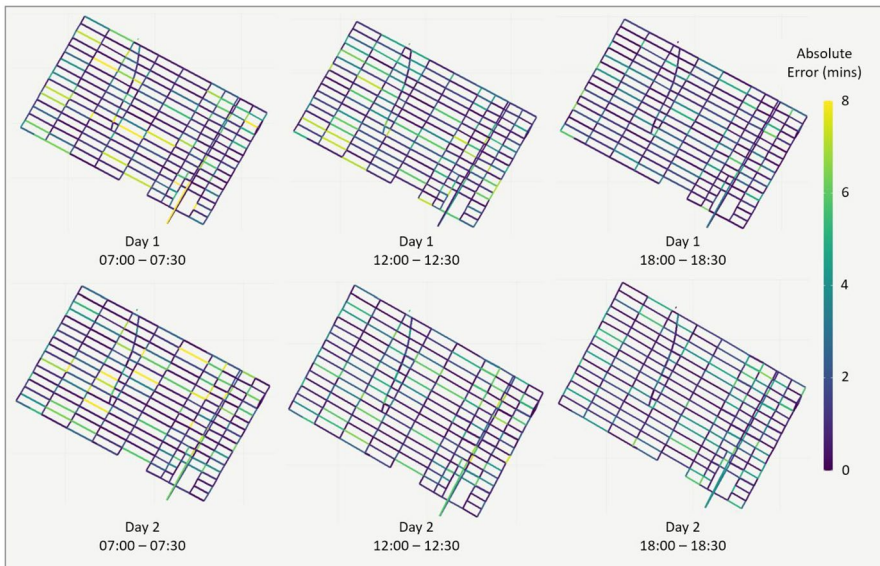


Fig. 14 Absolute Error in 6 Instances

After applying Zhan et al. (2013)'s methodology to these instances, we obtain the link travel time and compare with our three-step framework's result.

Figure 14 shows the absolute error between our result and Zhan et al. (2013)'s at these 6 instances. The links are color-coded based on this absolute error with darker color indicating lower error and vice versa and we notice the following pattern. First, the absolute error is highest in the first time period of 07:00–07:30 and it decreases in the noon and evening time periods. Since later time periods have more taxi trip records than the first time period, the accuracy of both methodologies are higher and there is less variance. However, the majority of the links are still within 2 min of absolute error indicating similar estimation. Second, links in the East–West direction have slightly more error than links in the North–South direction because of two reasons. First, North–South links are usually shorter than East–West and thus the error is smaller. Second, taxi demand going in the general North–South of Manhattan (e.g., from Central Park to Financial District) is proportionally higher than the demand going East–West. With more taxi trips recorded in the North–South direction, the estimation of North–South links for both methodologies will be more accurate and stable, resulting in less error.

Next, we use the link travel time from both methodologies combined with the predicted path choice from step 1 to calculate the predicted trip travel time and validate with the observed trip travel time. Table 3 shows the Mean Absolute Error (MAE) and the Mean Absolute Percentage Error (MAPE) of these two methodologies at 6 instances. In terms of, our three-step framework is slightly more accurate across all 6 instances, but two methods still have very low error at an average of 1.5 and 1.0 min respectively. In terms of MAPE, our framework also has lower error except for the fourth instance. On average, the MAPE for Zhan et al. (2013) and our

Table 3 Validation between Observed and Predicted Trip Travel Time

Date	Time	Mean Absolute Error (minutes)		Mean Absolute Percentage Error (%)	
		Zhan et al. (2013)	Three-steps Framework	Zhan et al. (2013)	Three-steps Framework
Day 1	07:00—07:30	1.4	1.0	27.9%	22.8%
	12:00—12:30	1.6	1.1	27.1%	24.3%
	18:00—18:30	1.1	1.0	29.1%	29.6%
Day 2	07:00—07:30	1.0	1.0	35.3%	29.8%
	12:00—12:30	2.2	1.1	23.7%	15.6%
	18:00—18:30	1.6	1.2	38.1%	33.1%

three-steps framework are 30% and 26% respectively. Zhan et al. (2013)'s framework requires to subset the taxi trip record of which origin and destination is within the studied network only. Since our Midtown network is small, the taxi trip is relatively short resulting in a shorter travel time and a higher value of MAPE.

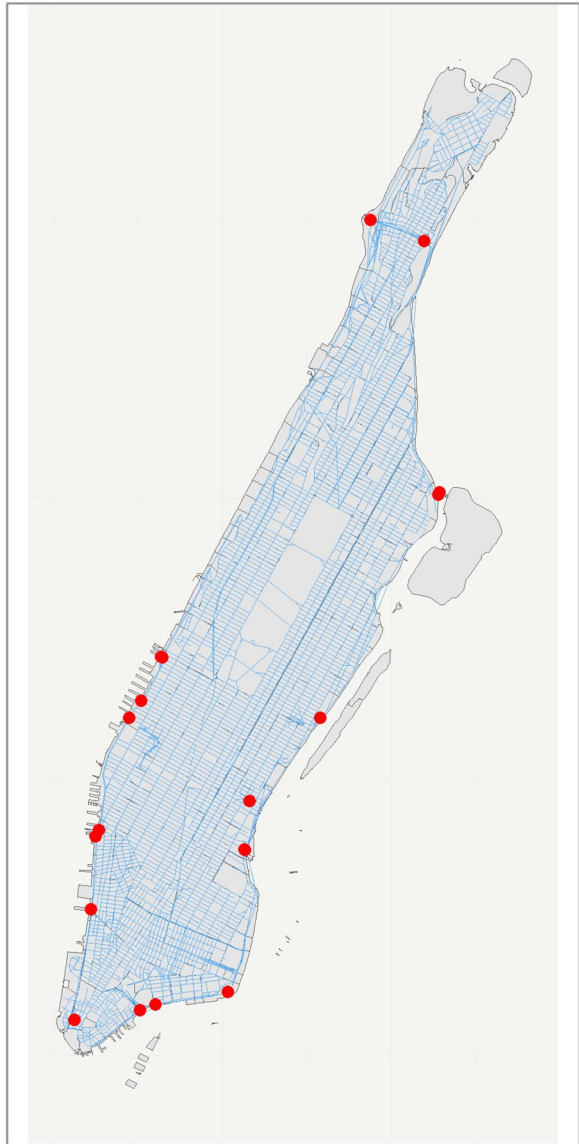
5.2.2 Validation with Ground Truth Speed Sensor

We obtain the ground truth speed data from the City of New York.³ The New York Department of Transportation's Traffic Management Center and other local agencies have installed multiple traffic speed detectors in a variety of form. One of which is camera detector which also provides live view of traffic and can be viewed via the website.⁴ Therefore, we filter the data based on the following criteria to ensure it is compatible for comparison with the framework's result: (1) the years are between 2014 and 2015, (2) weekday is Monday and time of the day is between 06:00 – 20:00, (3) the traffic speed detectors fall within the Manhattan borough only. Ultimately, we get 19 traffic speed detectors and a total of 57,611 observations from 4/18/2015 – 11/30/2015. The location of these speed detectors is shown in Fig. 15 below:

Each speed detector is assigned to the corresponding link based on its location and the traffic flow direction of which the camera is capturing. The timestamp of each observation is also translated into the equivalent time period so that the ground truth data and model's result are comparable. We compute the absolute error between the model's result and the ground truth data, aggregate by time period, and utilize boxplot to show the distribution of these errors as a box plot at each time period as shown in Fig. 16. The majority of these boxplots have median typically falling under 5 which indicates the framework has high level of accuracy. The 25th percentile is close to zero whereas the 75th percentile is around 10 mph. However, the number of outliers are substantial and the

³ <https://data.cityofnewyork.us/Transportation/Real-Time-Traffic-Speed-Data/qkm5-nuaq>

⁴ <https://webcams.nyctmc.org/>

Fig. 15 Speed Camera Location

distribution of which is wide. We observe that the time period 5–7 and 19–21 or correspondingly 08:00 – 09:30 and 15:00 – 16:30 have more outliers than the rest. During these hours, the network usually experiences traffic congestion, and the travel speed would vary greatly from minutes to minutes. Our model is estimating a 30-min average and these variations result in a high number of outliers. Beside congestion, some driver tends to drive leisurely slow and chooses longer but less traffic route in contrast to rushing driver choosing shortest path. These are heterogeneity in driving behavior, and it can increase the framework's error.

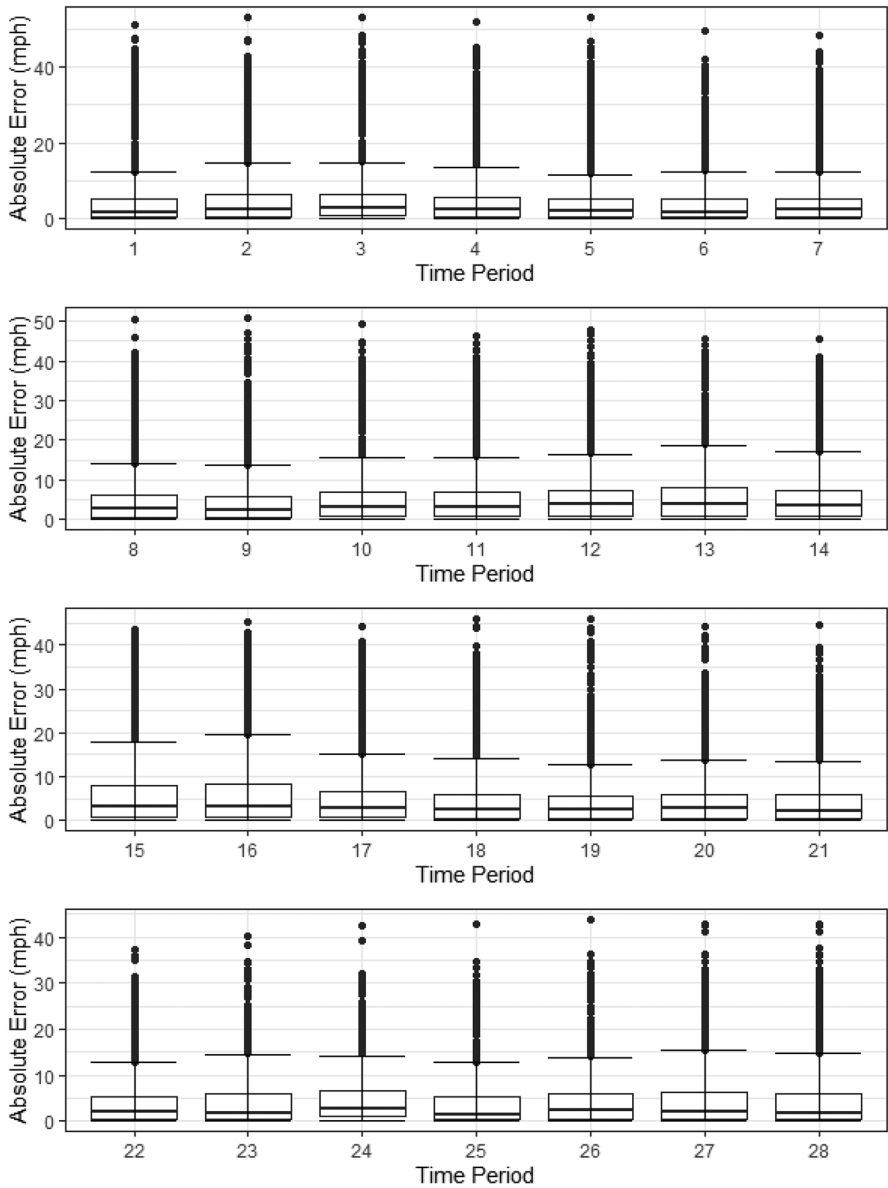
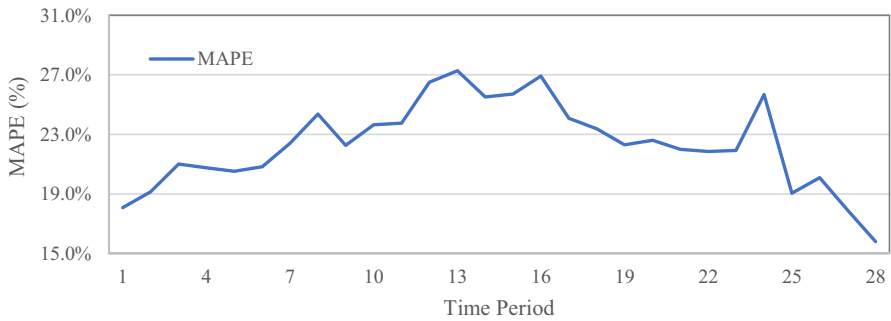
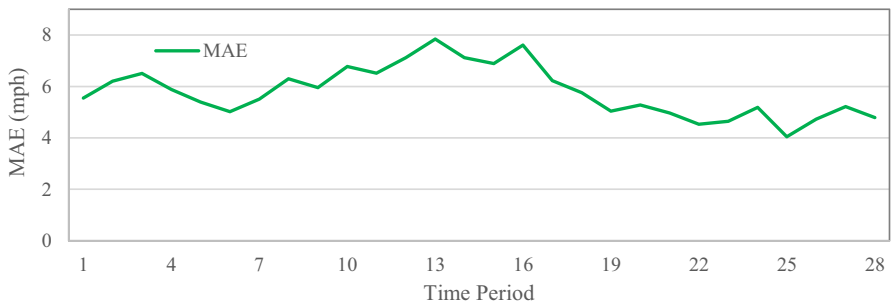


Fig. 16 Boxplot of Absolute Error between Framework’s Result and Ground Truth

Finally, as previously mentioned in step 2 evaluation section, the framework does not capture intersection delay and the result tends to overestimate the individual link travel time as, which ultimately contributes to the absolute error. However, giving only limited knowledge (i.e., sparse taxi trip), the framework is still able to estimate travel speed within a 5-mph accuracy.



(a) Mean Absolute Percentage Error



(b) Mean Absolute Error

Fig. 17 Framework's MAPE and MAE by Time Period

In addition to Fig. 17 showing the distribution error, we also provide the Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE) by each time period in Fig. 17. The MAPE averages around 22.3% and the values are slightly higher during midday compared to the early morning or late afternoon. The MAE plot also tells the same story where higher values are observed during midday. As shown in Fig. 8b, the number of taxi trips during midday is significantly less than the morning peak hours or later in the evening where people tend to go out for recreational purposes. Therefore, the input for step 3 of during these hours are not as strong the others and as a result, the error is higher. However, the average MAE of all time period is still 5.8 mph indicating reasonable estimation accuracy.

6 Conclusions and Future Work

In this paper, we propose a sequential three step framework to solve the problem of Network Wide Dynamic Link Travel Speed using only the Taxi Trip Dataset. In the first two steps, information from taxi trip record such as pickup and drop-off location, travel time, and distance are processed to give output of partial link travel speed. The third step makes use of a novel deep learning model which consists of two main components which are Traffic Graph Convolution (TGCN) for

capturing spatial relation and $TGCN_{lstm}$ for temporal relation. The result suggests that the framework can estimate dynamic link level travel speed in dense urban area and the accuracy is high as shown in the framework's application and evaluation in the New York case study. The significance of our study are (1) it only requires one open source (or low-cost) and passively collected dataset (i.e., Taxi); (2) unlike other normal GCN, the TGCN is specifically designed for traffic network and is able to capture directional flow of traffic through the Node-Link Embedding and Modified Adjacency Matrix; (3) in TGCN-LSTM, the model encourages node to take both historical data from itself and its neighbor; and (4) the model is capable of obtaining network wide travel speed for bigger networks with up to 9,500 links.

However, the three-step framework has several drawbacks. First, the computational efficiency in the first step is low since for each taxi trip record, the following two tasks must be executed: coordinate mapping and kth-shortest-path generation. We make full use of our computer's capability through parallel computing and efficient algorithm design. The process is accumulative and can be solved independently, but it still demands time due to the shear amount of data (i.e., up to one week). Second, the Yen's Algorithm generates an alternative path set that lacks diversity and cannot capture real-world driver's route choice. Third, we cannot accurately evaluate step 1 and 2. In step 1 the ground truth data on the taxi's path and itinerary are unknown whereas step 2 can overestimate link travel time due to unknown in intersection delay. For future research, all of these drawbacks can be fully addressed if taxi trips GPS traces (i.e., not just origin and destination but location at different timestamp) is made available.

A possible approach to deal with the estimation of path travel times from pick-up and drop-off time stamps lies with consideration of delay in the presence of traffic control devices such as signalized or unsignalized intersections, is to have detailed attributes of type of signal (signalized, stop or yield sign), and signal characteristics (timing of pre-timed signalized, actuated-signalized, semi-actuated signalized). The methodology proposed in the paper can incorporate the effect of signals in path travel time estimation, though in the case study in the absence of such data the delay accounted from signals are not considered. Considering dynamic state of traffic flow and signal timings are not always available on the urban road network and the goal of this paper is develop a model to estimate travel time as accurate as possible with least information and make the model more generic for planning applications for large networks, the results presented in the case study may associate with under or over estimation of path travel times.

We believe our research can be beneficial for urban planners in the ITS domain, especially those in developing economies without state-of-the-art infrastructure already in place. Future work for this study can concentrate on integrating the travel time estimation not only of road network but also other modes of travel especially public transportation including transit and bike providing urban dwellers complete information on travel cost between competitive travel modes for their decision making. In addition, further investigation could be emphasized on computational efficiency and intra-direction convolution, which allows for not only through movement but also turning movements.

Appendix A. Sets, Parameters, and Decision Variables used in the paper

Sets

N Set of nodes

A Set of links

F Set of node features or directions

K Set of layers

I Set of taxi trips

T Set of time periods

Parameters

η Total number of trips selected for subset S in the TTS model of step 2

L Total number of links included in subset S in the TTS model of step 2

P_i $[A \times 1]$ vector representing links comprising the travel path of taxi trip i

β Hyperparameter in the TTS model of step 2

$length_a$ Length of link a

t_i^{pred} Predicted travel time for taxi trip i

t_i^{obs} Observed travel time for taxi trip i

Δ_i Intersection delay of trip i

α_a^{link} North-bearing angle of link a

A Modified directional adjacency matrix with dimension $[N \times N \times F]$

a^f Subset of A at f with dimension $[N \times F]$

H_k^u Feature representation of a node u at k^{th} hidden layer

H_k^t $[N \times F]$ matrix contains information of all nodes in the network at k^{th} hidden layer and time period t

for_t Forget value for $\text{TGCN}_{\text{Istm}}$ at time period t

inp_t Input value for $\text{TGCN}_{\text{Istm}}$ at time period t

Out_t Output value for $\text{TGCN}_{\text{Istm}}$ at time period t

act_t Activation value for $\text{TGCN}_{\text{Istm}}$ at time period t

s_t Internal State value for $\text{TGCN}_{\text{Istm}}$ at time period t

$Loss$ Overall loss value for the deep learning model in step 3

Decision Variables

$x_i = \{0, 1\}$ 1 if trip i is selected for subset S in the TTS model of step 2, and 0 otherwise

tt $[A \times 1]$ vector representing link travel time from PLTT model

W^t $[N \times N]$ weight matrix in TGCN for passing information between neighbors at time period t

$W^{\text{for}}, W^{\text{inp}}, W^{\text{act}}, W^{\text{out}}$ $[N \times 2N]$ weight matrices for passing information and calculating internal state values in $\text{TGCN}_{\text{Istm}}$ cell

$B^{\text{for}}, B^{\text{inp}}, B^{\text{act}}, B^{\text{out}}$ $[N \times F]$ Bias matrices in $\text{TGCN}_{\text{Istm}}$ cell

Models and Operations

PCP Path Choice Prediction. The first step in the framework

PLTT Partial Link Travel Time Prediction. The second step in the framework

TTS Taxi Trip Subsetting. An inner preprocessing model within step 2

TGCN Traffic Graph Convolution Network. An operation for passing message between host node and its neighbor, specifically design to capture directional traffic flow

TGCN_{Istm} Traffic Graph Convolution Network with Long-Short Term Memory. An operation for passing message between host node's historical data as well as its neighbor's

Appendix B

Yen's Algorithm
Input: Graph $G(V, E)$; Origin o ; Destination d ; number of shortest path K
Output: A set of alternative paths $A = [p_1, \dots, p_k]$
1 Find the first shortest path: $A[0] = \text{Dijkstra}(G, o, d)$;
2 Initialize a container for storing potential paths: $B = []$
3 For $k = 1$ to K do
4 A spur node ranges from the first to the node next to last of the previous k -shortest-path
5 For $i = 0$ to $\text{length}(A[k-1]) - 2$ do
6 Assign the spur node: $V_{\text{spur}} = A[k-1].\text{node}(i)$
7 Create root path by taking nodes from origin to spur node: $P_{\text{root}} = A[k-1].\text{nodes}(0, i)$;
8 For every path p in A do :
9 If $P_{\text{root}} == p.\text{nodes}(0, i)$
10 Remove the edges to ensure the spur path is different than the previous k -shortest-path: remove $p.\text{edge}(i, i+1)$ from G
11 End p loop
12 Remove nodes from G that are the same as root path except for spur node: remove $P_{\text{root}}[-V_{\text{spur}}]$ from G ;
13 Create spur path from spur node to destination: $P_{\text{spur}} = \text{Dijkstra}(G, V_{\text{spur}}, d)$;
14 A complete path is a combination of root and spur path: $P_{\text{total}} = P_{\text{root}} + P_{\text{spur}}$
15 Add the potential path to the storage: $B.\text{append}(P_{\text{total}})$
16 End i loop
17 Sort the potential paths from smallest to largest cost: $B.\text{sort}()$
18 The k th-shortest path is the path with lowest cost from B : $A[k] = B[0]$
19 End k loop
20 Return A

Acknowledgements This research is partly funded by the Center for Transportation Innovations in Education and Research (C-TIER) at the University of Memphis. The authors are grateful to associate editor and anonymous reviewers for their constructive feedback. Any opinions, findings, and recommendations expressed by the authors herein do not reflect the view of the agency funding and supporting this study.

Data Availability The data used in this study are publicly available. The New York Road Network dataset is provided by OpenStreetMap via the following link: <https://www.openstreetmap.org/relation/175905>. The New York Taxi dataset is made available by the New York City Taxi and Limousine Commission at <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>. The exact algorithm for the framework can be provided upon request at hngo@memphis.edu.

Declarations

Conflict of Interest All authors declare that they have no conflicts of interest.

References

- Biem A, Bouillet E, Feng H (2010) IBM infosphere streams for scalable, real-time, intelligent transportation services. Proc 2010 ACM SIGMOD Int Conf Manag Data [WWW Document]. <https://doi.org/10.1145/1807167.1807291>. (Accessed 20 Jul 2020)
- Chondrogiannis T, Bouros P, Gamper J, Leser U, Blumenthal DB (2020) Finding k-shortest paths with limited overlap. VLDB J 29:1023–1047. <https://doi.org/10.1007/s00778-020-00604-x>
- City of New York (2019) Taxi Trips Record from the New York City Taxi & Limousine Commission [WWW Document]. <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>. (Accessed 20 Jul 2020)
- Cui Z, Henrickson K, Ke R, Wang Y (2019) Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. IEEE Trans Intell Transp Syst 21(11):4883–4894
- Diao Z, Wang X, Zhang D, Liu Y, Xie K, He S (2019) Dynamic Spatial-Temporal Graph Convolutional Neural Networks for Traffic Forecasting. Proc AAAI Confer Artif Intel 33:890–897. <https://doi.org/10.1609/aaai.v33i01.3301890>
- DOT (2019) Intelligent Transportation Systems - Benefits of Intelligent Transportation Systems Fact Sheet [WWW Document]. https://www.its.dot.gov/factsheets/benefits_factsheet.htm. (Accessed 20 Jul 2020)
- Duan Y, Lv Y, Liu Y-L, Wang F-Y (2016) An efficient realization of deep learning for traffic data imputation. Transport Res C: Emerg Technol 72:168–181. <https://doi.org/10.1016/j.trc.2016.09.015>
- Ermagun A, Levinson D (2018) Spatiotemporal traffic forecasting: review and proposed directions. Transp Rev 38:786–814. <https://doi.org/10.1080/01441647.2018.1442887>
- Fountoulakis M, Bekiaris-Liberis N, Roncoli C, Papamichail I, Papageorgiou M (2017) Highway traffic state estimation with mixed connected and conventional vehicles: Microscopic simulation-based testing. Transport Res C: Emerg Technol 78:13–33. <https://doi.org/10.1016/j.trc.2017.02.015>
- Frank M (1981) The Braess paradox. Math Program 20:283–302. <https://doi.org/10.1007/BF01589354>
- Geng X, Li Y, Wang L, Zhang L, Yang Q, Ye J, Liu Y (2019) Spatiotemporal Multi-Graph Convolution Network for Ride-Hailing Demand Forecasting. Proc AAAI Confer Artif Intell 33:3656–3663. <https://doi.org/10.1609/aaai.v33i01.33013656>
- Google (2020) TensorFlow [WWW Document]. TensorFlow. <https://www.tensorflow.org/>. (Accessed 23 Sept 2020)
- Hochreiter S, Schmidhuber J (1997) LSTM can Solve Hard Long Time Lag Problems. In: Mozer MC, Jordan MI, Petsche T (eds) Advances in Neural Information Processing Systems 9. MIT Press, pp 473–479
- Hoffman W, Pavley R (1959) A Method for the Solution of the Nth Best Path Problem. J ACM 6:506–514. <https://doi.org/10.1145/320998.321004>
- Hu X, Chiu Y-C (2015) A Constrained Time-Dependent K Shortest Paths Algorithm Addressing Overlap and Travel Time Deviation. Int J Transport Sci Technol 4:371–394. [https://doi.org/10.1016/S2046-0430\(16\)30169-1](https://doi.org/10.1016/S2046-0430(16)30169-1)
- Jenelius E, Koutsopoulos HN (2013) Travel time estimation for urban road networks using low frequency probe vehicle data. Transport Res B: Methodol 53:64–81. <https://doi.org/10.1016/j.trb.2013.03.008>
- Kawasaki Y, Hara Y, Kuwahara M (2019) Traffic State Estimation on a Two-Dimensional Network by a State-Space Model. Transport Res Proc 38:299–319. <https://doi.org/10.1016/j.trpro.2019.05.017>
- Ke J, Zheng H, Yang H, Chen X((2017) Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. Transport Res C: Emerg Technol 85:591–608. <https://doi.org/10.1016/j.trc.2017.10.016>
- Khan SM, Dey KC, Chowdhury M (2017) Real-Time Traffic State Estimation With Connected Vehicles. IEEE Trans Intell Transp Syst 18:1687–1699. <https://doi.org/10.1109/TITS.2017.2658664>
- Kipf TN, Welling M (2017) Semi-Supervised Classification with Graph Convolutional Networks. <https://arxiv.org/abs/1609.02907>

- Kumar A, Haque K, Mishra S, Golias MM (2019) Multi-criteria based approach to identify critical links in a transportation network. *Case Stud Transport Pol* 7(3):519–530
- Lin L, He Z, Peeta S (2018) Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transport Res C: Emerg Technol* 97:258–276. <https://doi.org/10.1016/j.trc.2018.10.011>
- Liu H, Jin C, Yang B, Zhou A (2018) Finding Top-k Shortest Paths with Diversity. *IEEE Trans Knowl Data Eng* 30:488–502. <https://doi.org/10.1109/TKDE.2017.2773492>
- Liu Z, Liu Y, Meng Q, Cheng Q (2019) A tailored machine learning approach for urban transport network flow estimation. *Transport Res C: Emerg Technol* 108:130–150. <https://doi.org/10.1016/j.trc.2019.09.006>
- Nantes A, Ngoduy D, Bhaskar A, Miska M, Chung E (2016) Real-time traffic state estimation in urban corridors from heterogeneous data. *Transport Res C: Emerg Technol, Adv Network Traffic Manag: from Dynamic State Estimation Traffic Control* 66:99–118. <https://doi.org/10.1016/j.trc.2015.07.005>
- Papageorgiou M, Kiakaki C, Dinopoulou V, Kotsialos A, Wang Y (2003) Review of road traffic control strategies. *Proc IEEE* 91:2043–2067. <https://doi.org/10.1109/JPROC.2003.819610>
- Polson NG, Sokolov VO (2017) Deep learning for short-term traffic flow prediction. *Transport Res C: Emerg Technol* 79:1–17. <https://doi.org/10.1016/j.trc.2017.02.024>
- Ralph J (2013) 2.5 quintillion bytes of data created every day. How does CPG & Retail manage it? [WWW Document]. IBM Consumer Products Industry Blog. <https://www.ibm.com/blogs/insights-on-business/consumer-products/2-5-quintillion-bytes-of-data-created-every-day-how-does-cpg-retail-manage-it/>. (Accessed 29 Dec 19)
- Sekula P, Marković N, Vander Laan Z, Sadabadi KF (2018) Estimating historical hourly traffic volumes via machine learning and vehicle probe data: A Maryland case study. *Transport Res C: Emerg Technol* 97:147–158. <https://doi.org/10.1016/j.trc.2018.10.012>
- Seo T, Bayen AM, Kusakabe T, Asakura Y (2017) Traffic state estimation on highway: A comprehensive survey. *Annu Rev Control* 43:128–151. <https://doi.org/10.1016/j.arcontrol.2017.03.005>
- Sheffi Y (1975) *Urban Transportation Networks* | Professor Yossi Sheffi [WWW Document]. <https://sheffi.mit.edu/book/urban-transportation-networks>. (Accessed 2 Mar 2020)
- Tan H, Feng G, Feng J, Wang W, Zhang Y-J, Li F (2013) A tensor-based method for missing traffic data completion. *Transport Res C: Emerg Technol* 28:15–27. <https://doi.org/10.1016/j.trc.2012.12.007>
- Thapa D, Paleti R, Mishra S (2022) Overcoming challenges in crash prediction modeling using discretized duration approach: An investigation of sampling approaches. *Accident Anal Prevent* 169:106639
- Wang Y, Zhang D, Liu Y, Dai B, Lee LH (2019) Enhancing transportation systems via deep learning: A survey. *Transport Res C: Emerg Technol* 99:144–163. <https://doi.org/10.1016/j.trc.2018.12.004>
- World Bank (2019) Urban population (% of total population) | Data [WWW Document]. URL <https://data.worldbank.org/indicator/SP.URB.TOTL.IN.ZS>. (Accessed 20 Jul 2020)
- Wu C, Thai J, Yadlowsky S, Pozdnoukhov A, Bayen A (2015) Cellpath: Fusion of cellular and traffic sensor data for route flow estimation via convex optimization. *Transport Res C: Emerg Technol* 59:111–128. <https://doi.org/10.1016/j.trc.2015.05.004>
- Xu D, Wei C, Peng P, Xuan Q, Guo H (2020) GE-GAN: A novel deep learning framework for road traffic state estimation. *Transportation Res C: Emerg Technol* 117:102635. <https://doi.org/10.1016/j.trc.2020.102635>
- Yang S, Ma W, Pi X, Qian S (2019) A deep learning approach to real-time parking occupancy prediction in transportation networks incorporating multiple spatio-temporal data sources. *Transport Res C: Emerg Technol* 107:248–265. <https://doi.org/10.1016/j.trc.2019.08.010>
- Yao H, Wu F, Ke J, Tang X, Jia Y, Lu S, Gong P, Ye J Li Z (2018, April) Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32, No. 1)
- Yen JY (1971) Finding the K Shortest Loopless Paths in a Network. *Manage Sci* 17:712–716
- Yeon J, Elefteriadou L, Lawphongpanich S (2008) Travel time estimation on a freeway using Discrete Time Markov Chains. *Transport Res B: Methodol* 42:325–338. <https://doi.org/10.1016/j.trb.2007.08.005>
- Yu B, Li M, Zhang J, Zhu Z (2019) 3D Graph Convolutional Networks with Temporal Graphs: A Spatial Information Free Framework For Traffic Forecasting. <https://arxiv.org/abs/1903.00919>
- Yu H, Wu Z, Wang S, Wang Y, Ma X (2017) Spatiotemporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks. *Sensors* 17:1501. <https://doi.org/10.3390/s17071501>

- Zhan X, Hasan S, Ukkusuri SV, Kamga C (2013) Urban link travel time estimation using large-scale taxi data with partial information. *Transport Res C: Emerg Technol* 33:37–49. <https://doi.org/10.1016/j.trc.2013.04.001>
- Zheng F, Van Zuylen H (2013) Urban link travel time estimation based on sparse probe vehicle data. *Transport Res C: Emerg Technol* 31:145–157. <https://doi.org/10.1016/j.trc.2012.04.007>
- Zhu L, Guo F, Polak JW, Krishnan R (2018) Urban link travel time estimation using traffic states-based data fusion. *IET Intel Transport Syst* 12:651–663. <https://doi.org/10.1049/iet-its.2017.0116>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.