A Comparative Evaluation of Established and Contemporary Deep Learning Traffic Prediction Methods

Ta Jiun Ting
Department of Mechanical and Industrial Engineering
University of Toronto, Toronto, Canada
Email: tajiun.ting@mail.utoronto.ca

Xiaoyu Wang
Department of Civil and Mineral Engineering
University of Toronto, Toronto, Canada
Email: cnxiaoyu.wang@mail.utoronto.ca

Islam Kamel
Department of Civil and Mineral Engineering
University of Toronto, Toronto, Canada
Email: islam.kamel@utoronto.ca

Scott Sanner
Department of Mechanical and Industrial Engineering
University of Toronto, Toronto, Canada
Email: ssanner@mie.utoronto.ca

Baher Abdulhai
Department of Civil and Mineral Engineering
University of Toronto, Toronto, Canada
Email: baher.abdulhai@utoronto.ca

Word Count: 6243 words + 5 tables (250 words per table) = 7493 words

Submitted [31 Jul 2020]

ABSTRACT

Traffic prediction is an essential component in intelligent transportation systems. Various methods have been developed to solve this challenging problem over the years, including time series models, regression models, and, more recently, deep learning models. This paper provides an unbiased comparison of these methods under a variety of settings and also addresses the critical question of whether deep learning approaches can offer significant improvements over classical machine learning methods. We used a traffic simulation model of the Greater Toronto Area to generate traffic data for a stretch of highway as well as an urban region. Using these datasets, we compared the methods under five scenarios with different prediction horizons, the presence of missing data, and the presence of traffic events unseen in the training data.

Our experimental results show that deep learning methods of traffic prediction, including graph convolutional neural networks, are effective for traffic prediction. Graph convolutional neural networks with shared parameters are very compact, resistant to overfitting, and performed well in all of our experiments. However, ensemble methods such as random forest regression can generate more accurate predictions at the cost of higher resource consumption during training, which may become a challenge in large transportation networks. Overall, we found that deep learning architectures should be carefully designed by restricting the input to features with known influences on the predictions, which can guide parameter learning and improve performance.

INTRODUCTION

Traffic congestion carries a significant cost to the regional economy. This cost can materialize in the form of wasted time and fuel by travelers, increased accident rates, and increased greenhouse gas emissions [1]. Due to insufficient land resources and the significant cost of building new roads, intelligent transportation systems (ITS) address this issue through effective traffic management aided by technology. Accurate prediction is essential in this strategy because it allows the system to respond proactively to the changing traffic demands.

Traffic is a highly-dynamic process evolving over space and time. For a stretch of highway, the traffic conditions at a given location influence its downstream sections since traffic moves forward. Meanwhile, the traffic conditions at a given location also influence its upstream sections as traffic congestion propagates backward via shock waves. These two basic processes are constantly present on any road network, evolving continually in response to the changes in traffic conditions. In an urban setting, the presence of intersections and traffic signals further influences the dynamics of traffic patterns and complicates the prediction problem.

Traffic prediction and modeling have been studied since the 1950s [2]. As computational and data collection technology improved, researchers applied increasingly complex methods to this problem including time series analysis, regression analysis, artificial neural networks, and most recently, deep learning (deep neural networks). However, there lacks a comprehensive evaluation of the different methods in different settings in the literature. This type of evaluation is important because it allows us to draw insight into the constitution of an accurate predictive method and incorporate proven techniques when developing new models.

This paper first outlines a few classes of existing methods and their innovations, then assesses their performance under a variety of both highway and urban scenarios. We evaluate the models using data from traffic simulations which avoids the problem of missing data due to sensor issues common in real-world datasets. In addition, the simulation environment allows us to easily adjust demands and road conditions to create a variety of benchmarks for the evaluation.

Problem Definition

Traffic prediction can be performed at the level of individual vehicles or the level of road links, which is a segment of a road. The prediction methods at the two different levels are very different. At the individual vehicle level, the predictor models driver behavior to predict the future trajectories of each vehicle. Meanwhile, at the link level, the predictor focuses on the macroscopic properties on each road and predicts their evolution over time. For this paper, we define the prediction problem as the link-level speed prediction problem.

The following notation is used throughout this paper:

- $\mathscr{G} = (\mathscr{V}, \mathscr{E})$: The directed graph which describes the road network. $\mathscr{V}$ is the set of links, and $|\mathscr{V}| = N$. $\mathscr{E}$ is the set of connections between links.
- $\mathscr{N}(i)$: The set of links that are immediate neighbors of link i. In other words, this set consists of all links that are connected directly with link i, and does not include link i itself.
- $\mathscr{M}(i)$: The set of links in the neighborhood of link i. This is not restricted to the immediate neighbors of link i, and also includes link i itself.
- $x_i^{(t)}$: A vector with length d. This represents the observation of link i at time t.
- $X^{(t)}$: A matrix with size $(N \times d)$. This represents the observation of the entire road network at time t.
- $\hat{x}_i^{(t)}$: A vector with length d'. This represents the prediction of link i at time t.

- $\hat{X}^{(t)}$: A matrix with size $(N \times d')$. This represents the prediction of the entire road network at time t.
- H: The prediction horizon.

Additionally, the variable i is reserved to distinguish a particular link while the variable j is used to distinguish a particular time slice.

The objective of a traffic prediction method is to generate accurate predictions of future traffic states by employing currently available information. This information may include past observations, road network characteristics, as well as any useful external information such as weather and the presence of traffic events. For this paper, we used past observations and the topology of the road network to create prediction models. Mathematically, the prediction problem can be described as learning a function f that maps the past observations to predictions using the graph $\mathscr{G}$ and minimize the prediction error L.

$$\hat{X}^{(t+1)}, \hat{X}^{(t+2)}, ..., \hat{X}^{(t+H)} = f(X^{(t)}, X^{(t-1)}, ..., \mathscr{G}) \tag{1}$$

$$L = \sum_{i=1}^{\mathscr{V}} \sum_{j=1}^{H} \parallel \hat{x}_i^{(t+j)} - x_i^{(t+j)} \parallel_2^2 \tag{2}$$

METHODS COMPARED

Autoregressive Models

We can view the macroscopic traffic properties on a link evolving over time as time series. Naturally, researchers have applied the techniques of time series analysis to traffic modeling and prediction. Autoregressive models for time series analysis considers each observation to be correlated with its previous observations. The autoregressive moving average (ARMA) model is a class of models that is extensively researched in this category.

The ARMA model combines the autoregressive component and the moving average component. The autoregressive component assumes the observation for link i at time t is a linear combination of past p observations on the same link, plus a constant and a white noise error term. This is described in Equation (3), where c is the constant, ε_t is the error term, and ψ_{ij} are model parameters. Similarly, the moving average component assumes the observation for link i at time t is a linear combination of past q error terms, plus the current white noise error term. This is illustrated in Equation (4), where ε_t represents the error term and ω_{ij} are model parameters. Finally, the full ARMA model combines the two component equations, shown in Equation (5). [3] describes the method to determine p and q, as well as calculating the model parameters.

$$\hat{x}_i^{(t+1)} = c + \sum_{j=0}^{p} \psi_{ij} x_i^{(t-j)} + \varepsilon_t \tag{3}$$

$$\hat{x}_i^{(t+1)} = \sum_{j=0}^{q} \omega_{ij} \varepsilon_{t-j} + \varepsilon_t \tag{4}$$

$$\hat{x}_i^{(t+1)} = c + \sum_{j=0}^{p} \psi_{ij} x_i^{(t-j)} + \sum_{j=0}^{q} \omega_{ij} \varepsilon_{t-j} + \varepsilon_t \tag{5}$$

From this class of autoregressive models, we chose to compare the autoregressive integrated moving average (ARIMA) model, which is extensively researched in traffic prediction [4, 5, 6, 7]. The ARMA model shown above describes a stationary stochastic process; however, traffic patterns often exhibit indications of non-stationarity and the ARIMA models apply an initial differencing step to eliminate the non-stationarity in the data. It should be noted that the ARIMA model remains a univariate model, thus road segments are decoupled from one another and analyzed individually through time without using information from nearby links. We used pmdarima [8] to construct the ARIMA model for this paper.

Regression Trees

Regression models predict future traffic properties by performing regression analysis on past observations. In the simplest case of linear regression, we can compute future state values as a linear combination of multiple explanatory variables, such as past observations of neighboring links. In contrast with the autoregressive models, regression models do not assume any relationship between the predicted variables (dependent variables) and the past observations (independent variables).

The regression tree model partitions the available data recursively based on the values of its independent variables, constructing a tree-like decision diagram. In its simplest form, it divides data into two partitions based on the value of a single independent variable. This is repeated until the desired number of leaf nodes is reached. At each leaf node of the tree, a simple model describes the data only within its partition. To generate predictions, the model traverses the regression tree based on the values of the independent variables. There is a test about the independent variables at each internal node of the tree, and the answer to the test determines which child branch should be visited next. Once the model reaches a leaf node, the model in the leaf node generates the dependent variables (prediction) based on the independent variables. Figure 1 is an example of a regression tree.

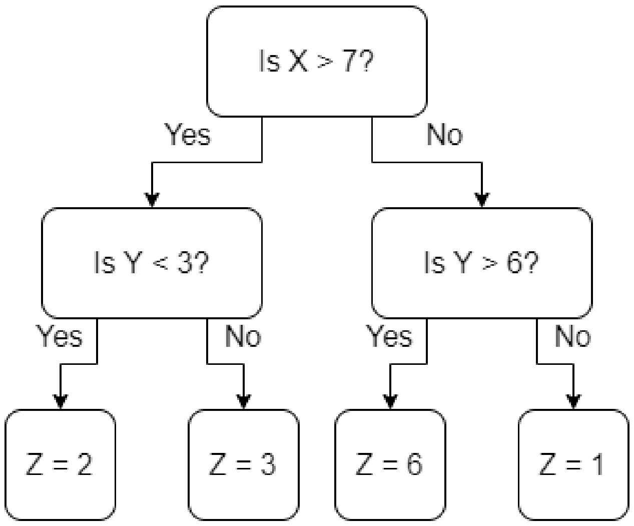

Figure 1 A simple regression tree. X and Y are the independent variables while Z is the dependent variable of this model.

The regression tree model can approximate complex functions using relatively simple functions with a sufficient number of decisions and leaf nodes. Typically, traffic prediction models of this category use the ensemble method that contains more than one regression tree. An example of the ensemble method is the gradient boosting method developed by Friedman [9], which constructs each subsequent tree with an emphasis on instances where the previously constructed trees performed poorly. This type of model is commonly cited as a powerful method for traffic prediction in the literature [10, 11].

We selected the random forest method from the class of regression trees in this paper [12]. Random forest splits the available data to create multiple regression trees during training, and averages the output of all trees during prediction. This creates a robust regression model that is highly resistant to overfitting, a common issue for regression trees. Since the independent variables are different for each link, a separate random forest is created. We constructed the random forest model using scikit-learn [13]. The independent variables to the model are the recent observations of the neighbors and the regression targets are the predictions $\hat{x}_i^{(t+1)}, \hat{x}_i^{(t+2)}, \ldots, \hat{x}_i^{(t+H)}$, this is identical to the linear regression model in equation (13). We omitted the mathematical formulation for this model due to its complexity and space limitations; details of this method can be found in [12].

Feedforward Neural Networks

Since the 1980s, the rise of artificial neural networks (and more recently, deep learning) have provided researchers with powerful new tools for traffic prediction. Feedforward neural networks (FNNs) are the first and simplest type of artificial neural network. The connections among the nodes do not form a cycle and information in this network travels only in the forward direction, from the input nodes to the output nodes. A feedforward neural network with a single hidden layer can approximate any continuous function in Euclidean space, making them universal approximators [14]. The notion of this network has been around for decades, but they require significant computational power and data to train (and tune). Therefore, its rise to prominence in research as well as application to traffic prediction is fairly recent with the advent of accelerated GPU-based computation as well as cloud-based computing.

In traffic prediction, traffic properties on a link are influenced by the recent states of nearby links, but the exact relationship is complex and unknown. Since a feedforward neural network can approximate complex functions, they are suitable for this task. The network can generate predictions for a link by using a combination of recent traffic states of neighbouring links and the target link itself as the input. Early adopters of this model in traffic prediction include [15, 16, 17, 18]. Recent advanced neural network training techniques [19, 20] have increased the depth of neural networks that can be practically built and trained; [21] provides an example of such a deep neural network used for traffic prediction. The potential of deep learning has also prompted the development of more complex neural network architectures such as the graph convolutional neural network that will be discussed and compared to in later sections.

For the class of feedforward neural networks, we created two models. The first model is the global model, which predicts for all links in the road network with a single neural network. Meanwhile the second model is the local model, which includes a separate neural network for each link to account for the different local traffic patterns of each road. All neural network contains one hidden layer, and the sizes of the hidden layer is the same for all links in the local model. The inputs to the neural networks are the recent observations of the entire network for the network-wide model, and recent observation of nearby links only for the local model. The two models and all subsequent neural network models used in this paper are built using PyTorch [22].

Mathematically, the global model can be defined as follows:

$$\begin{aligned} f &= \mathop{\Big|\Big|}_{j=0}^{p} \operatorname{vec}(X^{(t-j)})^\top \\ k &= \sigma(w_1^\top \cdot f^\top + b_1) \\ o &= \mathop{\Big|\Big|}_{j=1}^{H} \operatorname{vec}(\hat{X}^{(t+j)})^\top \\ &= (w_2^\top \cdot k + b_2)^\top \end{aligned} \tag{6}$$

Meanwhile, the local model can be defined as follows:

$$\begin{aligned} f_i &= \Big\|_{n \in \mathscr{M}(i)} \Big\|_{j=0}^{p} x_n^{(t-j)\top} \\ k_i &= \sigma(w_{i1} \cdot f_i^\top + b_{i1}) \\ o_i &= \Big\|_{j=1}^{H} \hat{x}_i^{(t+j)\top} \\ &= (w_{i2}^\top \cdot k_i + b_{i2})^\top \end{aligned} \tag{7}$$

where f, k, and o are respectively the input, hidden, and output feature vectors, while w_1, w_2, b_1, b_2 are model parameters; $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix, and $\|$ denotes the concatenation operator. Furthermore, p denotes the length of past observations to include in the feature vector and σ denotes the activation function, which are tuned hyperparameters of the model. Lastly, in the local model, the subscript i signifies that the vectors and model parameters are unique to each link.

Recurrent Neural Networks

In contrast to feedforward neural networks, a recurrent neural network (RNN) is designed to process sequential data. In an RNN, the connections of successive layers of nodes form a temporal sequence. Additionally, the parameters in the model are shared among time steps and the output of each layer is used as the input to the next, repeated until the end of the sequence. As such, this architecture can process sequential data of different lengths provided that the feature dimensions are constant for every entry in the sequence. Furthermore, an RNN can generate a sequence of outputs by continuously appending the latest output to the input sequence.

Basic RNNs are essentially deep feedforward neural networks with shared weights across time steps and a depth that increases with the sequence length of the data. This depth causes the vanishing gradient problem to arise during training [23]. The vanishing gradient problem is an issue encountered when training an artificial neural network using backpropagation, where training the parameters in the early layers of the neural network becomes difficult as the gradient becomes vanishingly small. The long short-term memory (LSTM) architecture is proposed to address this issue by incorporating gates to the network to limit information propagation [24], which is later simplified by the gated recurrent unit (GRU) architecture [25].

Recurrent neural networks have been actively used in traffic prediction research to capture the temporal dynamics of evolving traffic since the work of [26] in 2015. This framework represents the recent history of traffic states as a sequence and uses an RNN to generate traffic state predictions. Later improvements to this model include stacking multiple RNNs [27] and incorporating spatial structure of the road network in the RNN [28].

While RNNs can be used as standalone traffic prediction models, they are ultimately designed to capture the dynamics within a sequence. However, we cannot easily represent the complex spatial influences within a traffic network as a sequence. Therefore, some newer prediction models use RNNs to capture the temporal patterns of traffic, but also include a separate module to capture the spatial dependencies within a traffic network, examples of this include the graph convolutional methods discussed in the following section.

Similar to feedforward neural networks, we also built both global and local recurrent neural networks. The global model uses the recent history of the entire road network to produce network-wide predictions. Meanwhile, the local model contains a separate RNN for each link, and only employs the local history as inputs to the RNN. In order to simulate traffic propagation spatially, the hidden state of each link is shared with nearby links for every time step in the sequence. This propagation procedure is outlined by [11], and the extent of propagation is a tuned hyperparameter of the model. The length of the input sequence and the sizes of the hidden state is the same for all locations in the local model.

Mathematically, the global model can be defined as follows:

$$\begin{aligned} k^{(s)} &= \begin{cases} \text{GRU}\left(k^{(s-1)}, \text{vec}(X^{(s)})\right) & s \le t \\ \text{GRU}\left(k^{(s-1)}, \text{vec}(\hat{X}^{(s)})\right) & s > t \end{cases} \\ \text{vec}(\hat{X}^{(s+1)}) &= w^\top k^{(s)} + b \qquad s = t, t+1, ..., t+H-1 \end{aligned} \tag{8}$$

Meanwhile, the local model can be defined as follows:

$$\begin{aligned} k_i^{(s)} &= \begin{cases} \text{GRU}_i\left(z_i^{(s-1)}, x_i^{(s)}\right) & s \le t \\ \text{GRU}_i\left(z_i^{(s-1)}, \hat{x}_i^{(s)}\right) & s > t \end{cases} \\ z_i^{(s)} &= k_i^{(s)} + \sum_{n \in \mathcal{N}(i)} \alpha k_n^{(s)} \\ \hat{x}_i^{(s+1)} &= w_i^\top k_i^{(s)} + b_i \qquad s = t, t+1, ..., t+H-1 \end{aligned} \tag{9}$$

where k and z are respectively the hidden vector before and after propagation, w, b are model parameters, and α is a hyperparameter that controls the amount of propagation. $\text{vec}(\cdot)$ denotes the vectorization of a matrix, and $\text{GRU}(\cdot)$ denotes recurrent neural network updating operation using the GRU updating rules. Similar to the previous section, the subscript i signifies that the vectors and model parameters are unique to each link.

Graph Convolutional Neural Networks

A convolutional neural network (CNN) is a deep learning method where the convolution operation is the primary means of propagating information from input features through stacked hidden layers. In addition, the parameters of the convolution operations within a hidden layer are shared, as opposed to the the parameter sharing between time steps in an RNN. Each convolution operation is restricted to a small region in the feature space known as the receptive field. The receptive fields from multiple convolution operations partially overlap to cover the entire feature space. As a result, the CNN excels at extracting simple local patterns and assembling them into increasingly complex features that are informative to the task.

The weight sharing of the convolution operation requires the receptive fields to be congruent with one another. This is typically a grid structure (e.g., over an image) where the receptive field can be translated freely in the feature space. Therefore, CNNs are commonly used for image processing since images possess a convenient grid structure. CNNs cannot be readily applied to a road network to capture its spatial dependencies since road networks generally do not have a perfect grid structure. However, graph convolutional networks [29] generalize the convolution operation to the graph domain. Since 2017, graph convolutional networks have been increasingly applied to traffic prediction with improved performance over recurrent neural networks and feedforward neural networks [30, 31].

Graph convolutional networks use the graph structure to define the receptive fields of the convolution operation, where node information is shared along edges to nearby nodes. The convolution operation aggregates the information available within the receptive field to produce a hidden representation for every node, which can then be used in subsequent convolutional layers or to produce outputs. Since each node in the graph may have an arbitrary number of neighbors, we need to specify a procedure to determine the receptive field of every node. There are many forms of the graph convolution operation, differing with one another regarding the receptive field definition. For example, diffusion-convolutional neural networks [32] model the propagation as a particle diffusion process and the diffusion probability determines how much each node shares information with nearby nodes. Meanwhile, PATCHY-SAN [33] uses a fixed receptive field similar to a CNN and a node selection process determines which neighboring nodes belong in the receptive field. Other definitions of graph convolution also exist in the literature, examples can be found in [34].

The graph convolution operation can capture the spatial dependencies of traffic, which is first proposed by [31]. This work integrates the diffusion convolution operation into an encoder-decoder recurrent neural network to capture the temporal dependencies of traffic, and the results shows a modest improvement on highway data over existing methods. Later, [35] improves this model by adding residual connections in the recurrent neural network. We can also cite [11, 36, 37] for using other forms of graph convolution in traffic prediction.

We picked the graph recurrent neural network (GRNN) model [11] to represent the class of graph convolutional networks. The GRNN model combines graph convolution with a recurrent neural network to capture both the spatial and temporal dynamics of traffic. The GRNN propagates the hidden states of every node to nearby nodes every time step, which reflects the spatial dependencies between nearby locations. Similar to other graph convolution network models, the parameters are shared between different locations and creates a compact model.

Mathematically, the GRNN model can be defined as follows:

$$\begin{aligned}
k_i^{(s)} &= \begin{cases} \mathrm{GRU}\left(z_i^{(s-1)}, x_i^{(s)}\right) & s \leq t \\ \mathrm{GRU}\left(z_i^{(s-1)}, \hat{x}_i^{(s)}\right) & s > t \end{cases} \\
z_i^{(s)} &= k_i^{(s)} + \sum_{n \in \mathcal{N}(i)} \alpha k_n^{(s)} \\
\hat{x}_i^{(s+1)} &= w_i^\top k_i^{(s)} + b_i \qquad s = t, t+1, \ldots, t+H-1
\end{aligned} \tag{10}$$

where k and z are respectively the hidden vector before and after propagation, w, b are model parameters, and α is a hyperparameter that controls the amount of propagation. $\mathrm{GRU}(\cdot)$ denotes recurrent neural network updating operation using the GRU updating rules. This model is very similar to the local RNN model in the previous section; however, the weights of the GRU are shared between different locations in this model.

EXPERIMENTAL SETUP

Model Selection

For the comparison, we selected one representative implementation from each class of method discussed in the previous section. In addition, we also include three simple models in the comparison to serve as baselines: a model that always predicts the arithmetic mean of past observed values, a model that uses the most recent observation as the prediction, and a linear regression model that uses recent local observations as independent variables. We used scikit-learn to build and train the linear regression model [13].

Mathematically, the constant model is defined by:

$$\hat{x_i}^{(s)} = b_i \quad \text{for} \quad s = t+1, t+2, \ldots, t+H \tag{11}$$

Similarly, the previous interval model is defined by:

$$\hat{x_i}^{(s)} = x_i^{(t)} \quad \text{for} \quad s = t+1, t+2, \ldots, t+H \tag{12}$$

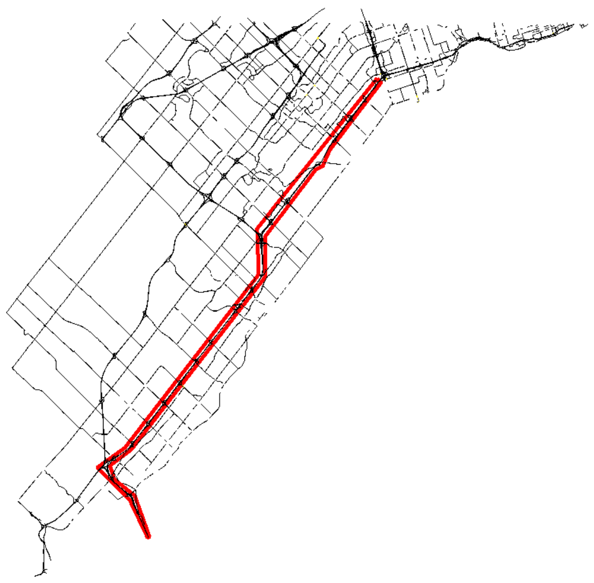

Figure 2 Map of the highway region chosen for this study.

Lastly, the linear regression model is defined by:

$$\begin{aligned} f_i &= \Big\Vert_{n \in \mathscr{M}(i)} \Big\Vert_{j=0}^{p} x_n^{(t-j)\top} \\ o_i &= \Big\Vert_{j=1}^{H} \hat{x}_i^{(t+j)\top} \\ &= (w_i^\top \cdot f_i + b_i)^\top \end{aligned} \tag{13}$$

where f and o are respective the input and output feature vectors, w, b are model parameters, and $\Vert$ denotes the concatenation operator. Similar to the previous section, the subscript i signifies that the vectors and model parameters are unique to each link.

Dataset

We procured two sets of data to represent the highway and urban traffic conditions; both sets are created using the Aimsun Next [38] simulation software as the data source. For the highway data, we used a simulation model of Queen Elizabeth Way, a highway in Ontario, Canada with 56 links on the eastbound direction of travel. For the urban data, we employed a simulation model of a 167-link region in downtown Toronto, Canada. The maps of the highway and urban regions are shown in Figure 2 and Figure 3, respectively.

The travel demand was collected from survey data in 2016 [39], then the simulation model was calibrated using measurements from the loop detectors installed along the road [40]. We built the simulation model using morning peak-hour travel demands, and each simulation is for the four-hour period between 6:00 and 10:00 AM. The speed (distance traveled per unit time) and flow (number of vehicles per unit time) for every link are extracted from the simulations in 1-minute intervals. Speed and flow are selected because they are the most common form of data in the real-world measured using loop detectors and GPS. To augment the simulation data, the simulation is run 50 times and each simulation uses the original travel demands multiplied by a random scalar factor between 0.5 and 1.5.

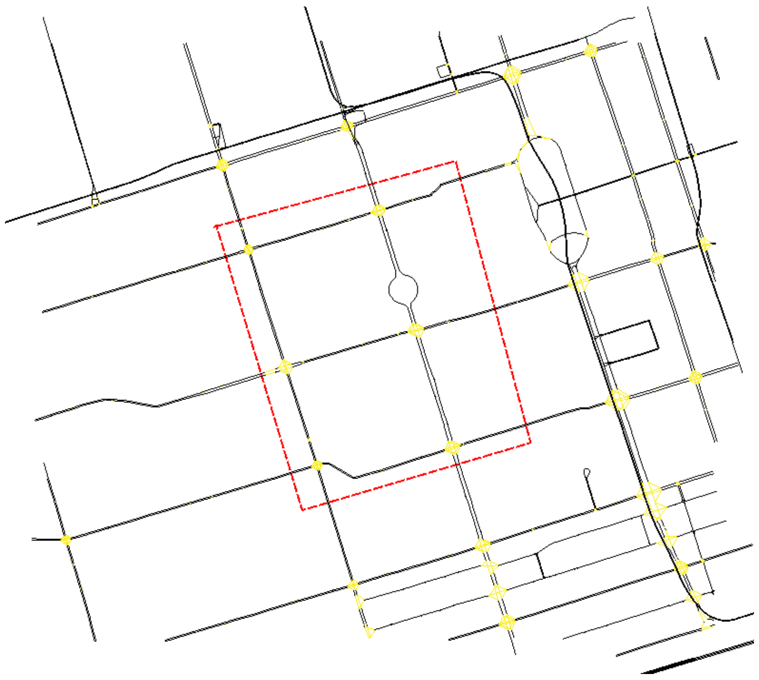

Figure 3 Map of the urban region chosen for this study.

Scenarios

In this paper, we assessed the models based on the predicted speed values in 5 different settings. For both the urban and highway dataset, we used a prediction horizon of 5 minutes as a base case. Missing data due to faulty sensors on the road is very common, thus we additionally included a scenario to evaluate the model performance under the presence of missing data in the highway setting. We chose 5% as the probability of missing data, evaluated using the same 5-minute prediction horizon. Meanwhile, we included a scenario with a 1-minute prediction horizon for the urban dataset, which is useful for tasks such as adaptive traffic signal control.

The last scenario evaluates the generalizability of each model. We generated an artificial dataset from the highway simulation model by blocking a lane in an arbitrarily chosen section; however, the same travel demands were used. This is intended to simulate traffic conditions in case of an event such as a traffic accident. We evaluated the models in this scenario by training them on the regular highway dataset, then the trained models were applied to this unseen artificial dataset to evaluate the performance metrics.

Evaluation Criteria

The goal of any predictive model is to minimize the difference between the predicted values and the actual values. To quantify this numerically, we use the following metrics to assess the performance: mean absolute error (MAE), mean absolute percentage error (MAPE), root-mean-square error (RMSE), and error variance. Given a sequence of H predictions $\{\hat{X}^{(t+1)}, \hat{X}^{(t+2)}, \ldots, \hat{X}^{(t+H)}\}$ and the actual observed value $\{X^{(t+1)}, X^{(t+2)}, \ldots, X^{(t+H)}\}$, MAE is the average of the absolute error while MAPE is the average of the absolute relative error. Alternatively, RMSE is the square root of the average squared errors, which is also the standard deviation of all errors. The calculation is shown below in equations (14), (15), and (16):

$$MAE = \frac{1}{|\mathscr{V}|}\frac{1}{H}\sum_{i=1}^{|\mathscr{V}|}\sum_{j=1}^{H}\left|x_i^{(t+j)} - \hat{x}_i^{(t+j)}\right| \tag{14}$$

$$MAPE = \frac{1}{|\mathscr{V}|}\frac{1}{H}\sum_{i=1}^{|\mathscr{V}|}\sum_{j=1}^{H}\left|\frac{x_i^{(t+j)} - \hat{x}_i^{(t+j)}}{x_i^{(t+j)}}\right| \tag{15}$$

$$RMSE = \sqrt{\frac{1}{|\mathscr{V}|}\frac{1}{H}\sum_{i=1}^{|\mathscr{V}|}\sum_{j=1}^{H}(x_i^{(t+j)} - \hat{x}_i^{(t+j)})^2} \tag{16}$$

RESULTS

The results of the models under the five scenarios are shown in the tables below. The mean and the confidence interval for each metric are calculated using 5-fold cross-validation. In 5-fold cross-validation, the full dataset is split into 5 partitions with equal sizes. In each iteration, one of the five partitions is reserved for testing while the remaining partitions are used for training; the process is repeated until each of the 5 partitions have been used as the test set. We did not perform cross-validation on the scenario with unseen artificial data since the test dataset is completely separated from training and validation datasets.

Except for the unseen test set scenario, the baseline models of constant and previous interval performed worse than every other model. However, the simple linear regression baseline proves itself to be comparable in performance with the deep learning architectures. The only scenario where the linear regression model had poor performance is under the presence of missing data. In general, the ARIMA model performed worse than the linear regression and random forest regression models. Overall, the random forest regression model performed the best in all scenarios and metrics.

Across all experiments, the local versions of the feedforward neural network and recurrent neural networks performed better than their global counterparts. In addition, the performance of the global RNN model is unstable sometimes as evidenced by the large spread of 95% confidence intervals. The local RNN performed better than its FNN counterpart for highway data, while their performances are relatively similar on the urban data. However, the FNN models are able to generalize better to unseen artificial test data. Finally, the graph neural network GRNN model performed competitively across all experiment scenarios, but it is consistently outperformed by the random forest model and the local RNN model.

Curiously, on the urban dataset, a few models are able to predict 5 minutes in advance better than 1 minute in advance. Similarly, on the highway data, the global RNN model performed better under the presence of missing data.

DISCUSSION

The ARIMA model is commonly used as a baseline model for comparison in the traffic prediction literature. However, our experiment results show that ARIMA performs only marginally better than just predicting the last observation. In addition, since there exists known influences among nearby roads, the univariate nature of ARIMA hampers its performance. This is especially evident as the prediction horizon increases, where evolving traffic patterns cause nearby information to become more important than local history. Overall, a good traffic prediction model should contain a more regional perspective rather than decoupling the roads from one another.

In the case of feedforward neural networks and recurrent neural networks, the local model performs better than the network-wide model. We speculate that this is because traffic dynamics are local, and using separate models for each location allows the parameters to be learned more easily. The global model has access to the recent history of the entire road network; therefore, the model can extract the relevant features to create accurate predictions for every location. On the other hand, each local model only has access to the recent history of a smaller nearby region. This restriction forces the local models to use features with

TABLE 1 Results for 5-minute prediction horizon on the highway dataset

		Constant	Previous Interval	Linear	ARIMA	Random Forest	Local FNN	Global FNN	Local RNN	Global RNN	GRNN
MAE (km/h)	Mean	33.16	4.56	3.29	4.41	2.76	3.41	3.80	2.86	7.11	3.30
	95% CI	32.34	4.24	3.02	4.12	2.52	3.27	3.41	2.66	4.28	2.77
		33.98	4.87	3.55	4.70	3.01	3.55	4.19	3.06	9.98	3.82
MAPE (%)	Mean	109.55	12.32	9.72	11.96	8.02	9.95	11.54	8.36	21.25	9.89
	95% CI	88.06	10.32	8.17	10.05	6.74	8.40	9.95	7.05	12.21	7.46
		131.04	14.32	11.28	13.87	9.31	11.50	13.13	9.67	30.29	12.33
RMSE (km/h)	Mean	35.46	9.34	5.12	9.13	4.74	5.37	5.76	4.83	11.81	5.37
	95% CI	34.72	8.72	4.62	8.53	4.15	5.01	5.05	4.33	7.56	4.44
		36.19	9.95	5.61	9.73	5.32	5.72	6.48	5.33	16.06	6.30

TABLE 2 Results for 5-minute prediction horizon on the highway dataset with 5% missing data

		Constant	Previous Interval	Linear	ARIMA	Random Forest	Local FNN	Global FNN	Local RNN	Global RNN	GRNN
MAE (km/h)	Mean	33.16	4.57	4.76	4.43	2.85	4.49	4.64	3.04	5.82	3.61
	95% CI	32.34	4.26	4.42	4.14	2.59	4.19	4.06	2.78	3.79	3.37
		33.98	4.88	5.10	4.72	3.11	4.78	5.21	3.30	7.85	3.84
MAPE (%)	Mean	109.55	12.36	12.55	12.00	8.26	12.29	13.39	8.88	16.78	10.91
	95% CI	88.06	10.37	10.89	10.09	6.96	10.45	11.61	7.35	11.73	8.77
		131.04	14.35	14.21	13.92	9.56	14.13	15.16	10.41	21.82	13.06
RMSE (km/h)	Mean	35.46	9.37	6.92	9.17	4.84	6.52	7.04	5.08	10.08	5.83
	95% CI	34.72	8.77	6.28	8.58	4.31	5.99	6.08	4.59	6.84	5.37
		36.19	9.97	7.56	9.77	5.38	7.05	8.01	5.58	13.32	6.29

TABLE 3 Results for 5-minute prediction horizon on the downtown dataset

		Constant	Previous Interval	Linear	ARIMA	Random Forest	Local FNN	Global FNN	Local RNN	Global RNN	GRNN
MAE (km/h)	Mean	6.68	6.53	4.30	4.87	3.94	4.42	4.43	4.18	4.27	4.35
	95% CI	6.64	6.52	4.28	4.83	3.91	4.38	4.20	4.11	4.15	4.32
		6.72	6.55	4.32	4.90	3.97	4.46	4.67	4.25	4.40	4.39
MAPE (%)	Mean	46.71	36.65	24.21	30.98	22.37	24.00	24.40	22.98	24.41	25.32
	95% CI	44.32	36.21	23.72	30.46	21.93	23.31	23.41	22.49	22.31	24.22
		49.11	37.09	24.69	31.50	22.80	24.68	25.40	23.46	26.52	26.42
RMSE (km/h)	Mean	10.04	10.88	6.61	7.73	6.30	6.92	6.87	6.77	6.82	7.01
	95% CI	9.91	10.85	6.56	7.69	6.26	6.81	6.47	6.56	6.63	6.94
		10.16	10.92	6.65	7.76	6.34	7.02	7.28	6.98	7.02	7.09

TABLE 4 Results for 1-minute prediction horizon on the downtown dataset

		Constant	Previous Interval	Linear	ARIMA	Random Forest	Local FNN	Global FNN	Local RNN	Global RNN	GRNN
MAE (km/h)	Mean	6.68	6.78	3.79	4.53	3.40	3.86	4.13	3.89	4.37	4.40
	95% CI	6.64	6.74	3.77	4.49	3.38	3.78	4.01	3.83	4.14	4.32
		6.72	6.81	3.80	4.57	3.42	3.93	4.24	3.94	4.59	4.47
MAPE (%)	Mean	46.71	38.05	21.48	28.19	19.14	20.10	22.38	21.18	26.60	26.58
	95% CI	44.32	37.74	21.12	27.70	18.90	19.26	22.02	20.82	24.33	25.27
		49.11	38.36	21.83	28.68	19.38	20.93	22.73	21.53	28.87	27.89
RMSE (km/h)	Mean	10.04	11.31	5.80	7.13	5.47	6.02	6.33	6.24	6.78	6.94
	95% CI	9.91	11.24	5.77	7.07	5.43	5.84	6.14	6.03	6.39	6.86
		10.16	11.38	5.83	7.18	5.50	6.20	6.52	6.45	7.16	7.02

TABLE 5 Results for 5-minute prediction horizon on the unseen artificial test set

	MAE (km/h)	MAPE (%)	RMSE (km/h)
Constant	31.01	96.08	33.75
Previous Interval	6.15	16.54	11.00
Linear	5.64	16.71	8.31
ARIMA	6.10	16.92	10.87
Random Forest	5.28	15.58	8.36
Local FNN	6.46	18.78	9.51
Global FNN	12.38	39.34	15.99
Local RNN	6.69	27.89	12.10
Global RNN	22.43	72.74	32.21
GRNN	5.70	15.61	8.99

known influences on the output, which guides parameter learning and creates more accurate predictions. Nevertheless, it is uncertain whether global models can achieve the same performance given access to more data.

Graph convolution networks can be very compact due to the shared parameters across different links, and our experiment results show that this can create a competitive model. However, removing parameter sharing can improve performance as evidenced by the better performance of the local RNN model. We believe this can be attributed to the different local dynamics of traffic, and having a custom updating operation for each location is essential in generating accurate predictions. Nevertheless, the reduced number of parameters of the GRNN helps prevent overfitting and generalize to unseen data, where it outperformed all other deep learning models in our experiment. One interesting research direction would be designing a graph neural network with a dynamic updating operation that accounts for the changing traffic dynamics between different roads and different time of day. This would allow the overall model to remain relatively compact while being expressive enough to represent all traffic patterns.

Across all scenarios, the random forest model generates the most accurate predictions. The ensemble nature of random forest makes it very robust to overfitting, and the model also generalizes well to unseen data (**Table 5**). However, due to its large number of parameters, it is not a compact model and uses significantly more memory compared to other models. In addition, since the regression features are different for every location, separate models need to be trained for every link, which slows model training and reduces scalability to larger networks.

Evaluation metrics such as MAE, MAPE, and RMSE demonstrate model performance in general; however, they cannot illustrate the finer trends of model performance. Therefore, we took the trained models and examined the model predictions at each location and identified two trends. On the highway dataset, the error on a link is negatively correlated with its length (**Figure 4**). We speculate that this is because shorter links are usually present at interchanges and exits, which influence traffic on the highway and increase the difficulty of prediction. Similarly, on the urban dataset, the error on a link is positively correlated to the its number of upstream and downstream connections (**Figure 5**), which means that links close to intersections have higher prediction errors. We speculate that this is because traffic signals have a greater influence on the conditions of these link rather than traffic propagation. Therefore, it would be interesting to include information of traffic signals into the predictive model and assess the performance.

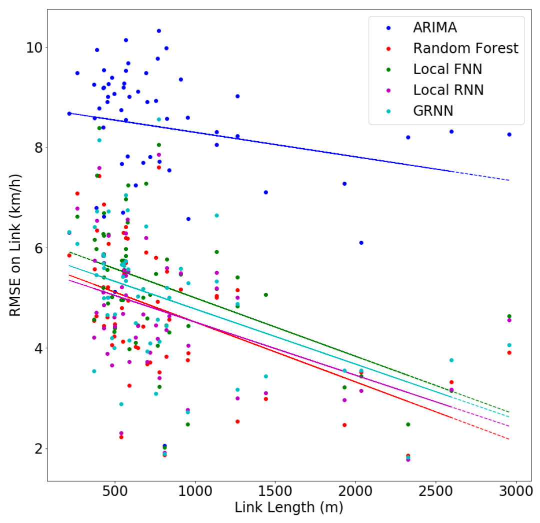

Figure 4 The RMSE of each link and the line of best fit for each model on the highway dataset.

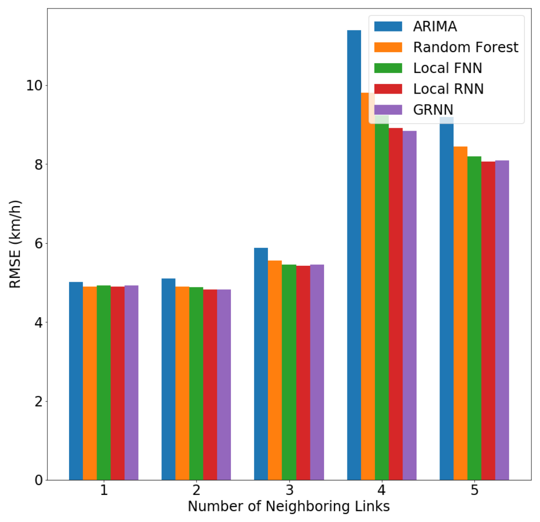

Figure 5 The RMSE of each model categorized by the number of neighboring links on the urban dataset.

CONCLUSION AND FUTURE WORK

In this paper, we compared different older and newer methods of forecasting short-term traffic in both the highway and urban settings using data from a traffic simulation software. The methods we included in this evaluation are ARIMA, random forest regression, fully-connected neural networks, recurrent neural networks, and the graph recurrent neural network model. Using the traffic model of the Greater Toronto Area, we generated a highway and an urban benchmark dataset. Finally, we created scenarios such as different prediction horizons and the presence of missing data to achieve a comprehensive comparison.

In our experiment, random forest regression consistently outperformed all other models, including existing deep learning methods. This suggests that existing models may be more susceptible to overfitting or that they are not expressive enough models for short-term traffic prediction. In addition, the results of our experiments show the importance of incorporating nearby observations in achieving more accurate predictions. However, it is also essential to limit the model inputs to only nearby observations since too many inputs obstruct model learning and deteriorate system performance. Finally, we also demonstrated the need for individual parameters for each prediction location due to the uniqueness of local traffic patterns. Overall, these facets should be carefully considered when developing new models for traffic prediction.

In our future work, we plan to include more models in this benchmark comparison. In particular, there has been much recent works on traffic prediction with graph convolutional neural networks. It may be interesting to assess whether the performance of graph recurrent neural networks are representative of the entire class of graph convolutional neural networks. In addition, we plan to augment our benchmark datasets with real-world data since simulation models cannot perfectly represent all real-world traffic conditions. Lastly, we plan to apply the findings of this paper to augment existing graph convolutional neural networks and create a model that captures the changing local traffic dynamics in all conditions.

ACKNOWLEDGMENTS

The authors would like to thank Huawei Canada Research Centre for financial & technical support.

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows: study conception and design: T. Ting, S. Sanner, B. Abdulhai; data collection: T. Ting, I. Kamel; analysis and interpretation of results: T. Ting; draft manuscript preparation: T. Ting, X. Wang, I. Kamel. All authors reviewed the results and approved the final version of the manuscript.

REFERENCES

[1] HDR Corporation. Costs of Road Congestion in the Greater Toronto and Hamilton Area: Impact and Cost Benefit Analysis of the Metrolinx Draft Regional Transportation Plan. Greater Toronto Transportation Authority; 2008.

[2] Lighthill MJ, Whitham GB. On kinematic waves II. A theory of traffic flow on long crowded roads. Proceedings of the Royal Society of London Series A Mathematical and Physical Sciences. 1955;229(1178):317–345.

[3] Box GE, Jenkins GM, Reinsel GC, Ljung GM. Time series analysis: forecasting and control. John Wiley & Sons; 2015.

[4] Ahmed MS, Cook AR. Analysis of freeway traffic time-series data by using Box-Jenkins techniques. 722; 1979.

[5] Moorthy C, Ratcliffe B. Short term traffic forecasting using time series methods. Transportation planning and technology. 1988;12(1):45–56.

[6] Lippi M, Bertini M, Frasconi P. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. IEEE Transactions on Intelligent Transportation Systems. 2013;14(2):871–882.

[7] Ma T, Zhou Z, Abdulhai B. Nonlinear multivariate time–space threshold vector error correction model for short term traffic state prediction. Transportation Research Part B: Methodological. 2015;76:27–47.

[8] Smith TG, et al.. pmdarima: ARIMA estimators for Python; 2017–. Accessed 28th June 2020. [Online]. Available from: `http://www.alkaline-ml.com/pmdarima`.

[9] Friedman JH. Greedy function approximation: a gradient boosting machine. Annals of statistics. 2001;p. 1189–1232.

[10] Yao H, Wu F, Ke J, Tang X, Jia Y, Lu S, et al. Deep multi-view spatial-temporal network for taxi demand prediction. In: Thirty-Second AAAI Conference on Artificial Intelligence; 2018. .

[11] Wang X, Chen C, Min Y, He J, Yang B, Zhang Y. Efficient metropolitan traffic prediction based on graph recurrent neural network. arXiv preprint arXiv:181100740. 2018;.

[12] Breiman L. Random forests. Machine learning. 2001;45(1):5–32.

[13] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.

[14] Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems. 1989;2(4):303–314.

[15] Dougherty MS, Cobbett MR. Short-term inter-urban traffic forecasts using neural networks. International journal of forecasting. 1997;13(1):21–31.

[16] Park B, Messer CJ, Urbanik T. Short-term freeway traffic volume forecasting using radial basis function neural network. Transportation Research Record. 1998;1651(1):39–47.

[17] Dia H. An object-oriented neural network approach to short-term traffic forecasting. European Journal of Operational Research. 2001;131(2):253–261.

[18] Abdulhai B, Porwal H, Recker W. Short-term traffic flow prediction using neuro-genetic algorithms. ITS Journal-Intelligent Transportation Systems Journal. 2002;7(1):3–41.

[19] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural computation. 2006;18(7):1527–1554.

[20] Bengio Y, Lamblin P, Popovici D, Larochelle H. Greedy layer-wise training of deep networks. In: Advances in neural information processing systems; 2007. p. 153–160.

[21] Lv Y, Duan Y, Kang W, Li Z, Wang FY. Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems. 2014;16(2):865–873.

[22] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems; 2019. p. 8026–8037.

[23] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks. 1994;5(2):157–166.

[24] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997;9(8):1735–1780.

[25] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:14061078. 2014;.

[26] Tian Y, Pan L. Predicting short-term traffic flow by long short-term memory recurrent neural network. In: 2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity). IEEE; 2015. p. 153–158.

[27] Cui Z, Ke R, Pu Z, Wang Y. Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. arXiv preprint arXiv:180102143. 2018;.

[28] Zhao Z, Chen W, Wu X, Chen PC, Liu J. LSTM network: a deep learning approach for short-term traffic forecast. IET Intelligent Transport Systems. 2017;11(2):68–75.

[29] Bruna J, Zaremba W, Szlam A, LeCun Y. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:13126203. 2013;.

[30] Yu B, Yin H, Zhu Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:170904875. 2017;.

[31] Li Y, Yu R, Shahabi C, Liu Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:170701926. 2017;.

[32] Atwood J, Towsley D. Diffusion-convolutional neural networks. In: Advances in Neural Information Processing Systems; 2016. p. 1993–2001.

[33] Niepert M, Ahmed M, Kutzkov K. Learning convolutional neural networks for graphs. In: International conference on machine learning; 2016. p. 2014–2023.

[34] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems. 2020;p. 1–21. Available from: `http://dx.doi.org/10.1109/TNNLS.2020.2978386`.

[35] Chen C, Li K, Teo SG, Zou X, Wang K, Wang J, et al. Gated Residual Recurrent Graph Neural Networks for Traffic Prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33; 2019. p. 485–492.

[36] Cheng X, Zhang R, Zhou J, Xu W. Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE; 2018. p. 1–8.

[37] Pan Z, Liang Y, Wang W, Yu Y, Zheng Y, Zhang J. Urban Traffic Prediction from Spatio-Temporal Data Using Deep Meta Learning. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '19. New York, NY, USA: Association for Computing Machinery; 2019. p. 1720–1730. Available from: https://doi.org/10.1145/3292500.3330884.

[38] Aimsun Next: Your Personal Mobility Modeling Lab;. Available from: https://www.aimsun.com/aimsun-next/.

[39] Ashby B. Transportation Tomorrow Survey 2016: Data Guide. Technical Report]. Available at http://dmg. utoronto. ca/pdf/tts/2016 . . . ; 2018.

[40] Kamel I, Shalaby A, Abdulhai B. Integrated simulation-based dynamic traffic and transit assignment model for large-scale network. Canadian Journal of Civil Engineering. 2019;(999):1–10.