Grids Versus Graphs: Partitioning Space for Improved Taxi Demand-Supply Forecasts

Neema Davis, Gaurav Raina, and Krishna Jagannathan

***Abstract*—Accurate taxi demand-supply forecasting is a challenging application of ITS (Intelligent Transportation Systems), due to the complex spatial and temporal patterns involved. We investigate the impact of different spatial partitioning techniques on the prediction performance of an LSTM (Long Short-Term Memory) network, in the context of taxi demand-supply forecasting. We consider two tessellation schemes: (i) the variable-sized Voronoi tessellation, and (ii) the fixed-sized Geohash tessellation. While the widely employed ConvLSTM (Convolutional LSTM) method can model fixed-sized Geohash partitions, the standard convolutional filters cannot be applied on variable-sized Voronoi partitions. To explore the impact of the Voronoi strategy, we propose the use of a GraphLSTM (Graph-based LSTM) model, by representing the Voronoi spatial partitions as nodes on an arbitrarily structured graph. The GraphLSTM model offers competitive performance against the ConvLSTM model, at a lower computational complexity, across three real-world large-scale taxi demand-supply data sets, with different performance metrics. To ensure superior performance across diverse settings, a HEDGE based ensemble learning algorithm is applied over the ConvLSTM and the GraphLSTM networks.**

***Index Terms*—Taxi demand-supply, spatial tessellation, time-series forecasting, ConvLSTM, GraphLSTM.**

I. Introduction

SPATIO-TEMPORAL forecasting has a wide range of applications, ranging from epidemic detection [1], energy management [2], to cellular traffic [3], among others. Location-based taxi demand and supply forecasting, one of the key components of ITS (Intelligent Transportation Systems), also relies heavily on accurate spatio-temporal forecasting. Mobility-on-Demand services such as e-hailing taxis, which have gained tremendous popularity in recent years, often face demand-supply imbalances. Mismatches frequently occur between the spatial distributions of the taxi demand and the available drivers, resulting in either scarcity or abundance of vacant taxis. For example, Fig. 1 presents a case of demand-supply mismatch near the city center in Bengaluru, India. We see that during the day hours, the high demand for taxis is met with inadequate supply. On the other hand, there is a surplus of vacant taxis during night hours, against low customer demand. Accurate demand-supply forecasts can

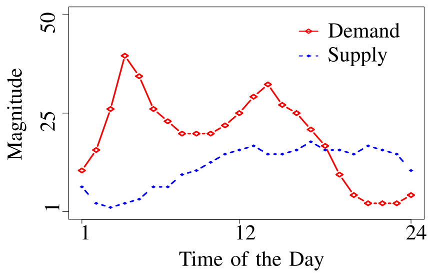


Fig. 1. The demand-supply patterns near Bengaluru city center averaged over Mondays across January-February 2016, where the mismatch between the demand for taxis and the available supply is visible. Our study is motivated by this demand-supply imbalance, which can be mitigated by accurate demand and supply forecasts.

mitigate this imbalance, thereby improving the efficiency of these taxi services. Information regarding the expected future demand and supply in a region can be used to re-route vacant cruising taxis, dynamically adjust the taxi fares, and recommend popular pick-up locations to the drivers.

A. Related Works

The recent popularity of e-hailing taxi services has generated substantial interest in developing efficient taxi demand-supply prediction algorithms [4]–[7]. In the past, taxi demand-supply prediction problems have mainly been formulated as classical time-series forecasting problems. The ARMA (Auto Regressive and Moving Average) family of time-series models have been applied to taxi demand prediction problems with satisfactory results; for example, see [8]. However, these time-series models rely on the stationarity assumption, which is often violated by real-world data. The capability of such classical methods to deal with high dimensional, complex, and dynamic time-series data is also limited. Meanwhile, the capabilities of NNs (Neural Networks) have inspired transportation researchers to leverage these ideas in the traffic forecasting domain with promising results [9], [10]. Traditional NNs cannot learn temporal dependencies, leading to the design of models that are more suited for sequence data such as RNN (Recurrent Neural Network) and its variants, namely LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) [11], [12]. Since the real-world data often exhibit both temporal and spatial variations, several spatio-temporal extensions of RNNs have been proposed. A widely employed extension in taxi data forecasting, known as the ConvLSTM (Convolutional LSTM) model, involves the addition of convolutional layers prior to the LSTM framework [16]. The standard convolutional layer can only be applied to a grid-structured

Manuscript received February 18, 2019; revised August 26, 2019; accepted April 13, 2020. The Associate Editor for this article was L. Yang. *(Corresponding author: Neema Davis.)*

The authors are with the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai 600 036, India (e-mail: ee14d212@ee.iitm.ac.in; gaurav@ee.iitm.ac.in; krishnaj@ee.iitm.ac.in).

Digital Object Identifier 10.1109/TITS.2020.2993798

input, to learn localized rectangular filters. This limits the application of the conventional ConvLSTM model to a city space partitioned into fixed-sized grids. Hence, a fixed-sized equally-spaced partitioning is often adopted in the spatio-temporal NN-based models for location-based taxi demand or supply forecasting [7], [17]–[20].

In our previous work [21], we explored a variable-sized partitioning scheme in addition to a fixed-sized scheme for taxi demand forecasting. Using classical time-series regression models, we observed a visible enhancement in the prediction performance with a variable-sized tessellation scheme in several scenarios. Data from the real-world often has a heterogeneous spatial distribution, which may not be captured faithfully with a fixed-sized partitioning scheme that is based on the assumption of spatial homogeneity. While the generalization capabilities of RNNs make them powerful tools for spatio-temporal modeling, assuming that the data is homogeneously distributed may limit their modeling capabilities. Hence, it is imperative to explore a variable-sized partitioning scheme in an RNN-based spatio-temporal modeling framework. Most of the currently popular spatio-temporal RNN models are based on ConvLSTM networks that are not suitable for modeling variable-sized partitioned spaces. Motivated by this observation, in this paper, we develop an LSTM framework that can extract the potential of variable-sized spatial partitions.

We consider two partitioning schemes in this study: (i) the Voronoi tessellation [22] that generates variable-sized partitions based on the nearest neighbor rule, and (ii) the Geohash tessellation [23] that generates partitions of fixed area using an alphanumeric encoding scheme. While dividing the city space into Voronoi partitions, we note that the arbitrarily spaced tessellations can be represented using graphs. The demand aggregated in each Voronoi partition can form a node in an arbitrary structured graph. Therefore, a Graph-based RNN holds potential in our scenario.

In the last couple of years, there has been substantial interest in devising Graph NNs, by extending the convolution operator to suit a more general graph-structured data [24]. In the context of traffic forecasting, Graph CNNs (Convolutional Neural Networks) have been applied to predict flows at traffic sensors [25]. By considering a road network as a graph and traffic sensors as nodes, Graph CNNs have been combined with RNNs to capture the spatial relationships between nodes [25]–[27]. However, there is limited research on incorporating graph RNNs in location-based taxi demand or supply forecasting. In [28], the authors do apply graphs to model non-euclidean correlations for ride-hailing demand forecasting, but the models are based on equally-spaced fixed-sized grid partitions. Essentially, the existing transportation literature on Graph NNs either consider traffic sensors as the graph nodes or learn graph-based correlations in a grid-partitioned space.

Our modeling framework deviates significantly from the existing literature as we employ a GraphLSTM (Graph-based LSTM) [26] model to learn an arbitrarily structured graph, where each node corresponds to the aggregated demand in a spatial Voronoi partition. To the best of our knowledge, in the context of spatial partitioning, Graph-based RNNs have not been explored in the literature. Another significant contribution of this paper lies in understanding the impact of different spatial partitioning schemes on the predictive performance of RNNs. To that end, we perform a comparison of the Geohash-based ConvLSTM and the Voronoi-based GraphLSTM models. These features set our work apart from the existing literature.[1]

B. Our Contributions

After the city is divided into fixed-sized rectangular cells and variable-sized polygon cells, we employ the standard ConvLSTM method to model the equally-spaced Geohash tessellated city and the GraphLSTM method to model the unequally-spaced Voronoi tessellated city. We compare their results with three baselines: (i) the vanilla LSTM based on Voronoi and Geohash schemes, (ii) the ARIMA (Auto Regressive Integrated Moving Average) model, and (iii) the ARIMAX (ARIMA with eXogenous inputs) model. When evaluated across three real-world data sets, the GraphLSTM model exhibits competitive prediction performance against the established baseline models, at a lower computational complexity. Further, we see that the prediction models exhibit non-stationary behavior, in addition to dependencies on the choice of the data set and performance metrics. To tackle this issue, we perform ensemble learning on the time-shifting models using an online non-stationary expert combining algorithm known as the dHEDGE [30]. By exponentially decaying the weights of the prediction models, the algorithm picks the best model for each time step in the forecasting horizon. Our main contributions are summarized below:

- This paper demonstrates the potential of Graph RNNs within a location-based spatial partitioning and forecasting framework.
- The GraphLSTM model offers competitive prediction performance at a lower computational complexity when compared with the ConvLSTM model, across diverse data sets using different performance metrics.
- The Voronoi-based GraphLSTM method outperforms the Geohash-based GraphLSTM and ConvLSTM methods in data scarce locations, and performs at least as well as others in data dense locations. This observation highlights the potential of irregular graphs in location-based forecasting.
- Applying the dHEDGE algorithm in conjunction with the ConvLSTM and GraphLSTM models ensures consistently superior prediction accuracy, across all the scenarios considered.

The rest of the paper is organized as follows. In Section II, we outline the problem statement. The spatial tessellation schemes are explained in Section III, along with a brief description of the data sets. In Section IV, the spatio-temporal models are discussed. The experimental settings and results are elaborated in Section V, followed by a description of the dHEDGE algorithm in Section VI. We conclude, with a summary of our results, in Section VII.

[1]A part of this work was presented as a poster at the NIPS Workshop on Machine Learning in Intelligent Transportation Systems, 2018 [29].

II. Problem Setting

For any location-based forecasting, the city space is tessellated into N regions. The set of regions $R = \{r_1, \ldots, r_N\}$ can be fixed-sized grids, variable-sized polygons, zip codes, etc. We consider the following: (i) fixed-sized rectangular grids called *geohashes*, and (ii) variable-sized polygon partitions called *Voronoi cells*. Let the demand and supply aggregated in every $r_i \in R$ form the sets $D = \{d_1, \ldots, d_N\}$ and $S = \{s_1, \ldots, s_N\}$. We assume that the data in the region of interest is related to its historical values and the values in its first-order neighboring regions. In this paper, our objectives are two-fold. First, given the set of all geohashes, our goal is to learn a function $\mathcal{F}(\cdot)$, mapping the demand (or supply) data in any geohash to its temporal and spatial neighbors. For the Geohash-based fixed-sized equally-spaced spatial structure, we use the ConvLSTM framework to learn a function $\mathcal{F}(\cdot)$ as:

$$\mathcal{F}([d_{1:t,1:N}]|geohashes) = [d_{t+1:t+h,1:N}],$$

where, h is the forecast horizon. Second, for exploring Voronoi-based variable-sized unequally-spaced spatial structure, we represent the tessellated city space by an undirected graph $\mathcal{G}$, where $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. The N regions will form a graph with $\mathcal{V}$ vertices and $\mathcal{E}$ edges. The connectivity between nodes is represented by an adjacency matrix $\mathcal{A} \in \mathbb{R}^{N \times N}$. The adjacency matrix is defined as follows:

$$\mathcal{A}_{i,j} = \begin{cases} 1 & \text{if there is a link connecting nodes } i \text{ and } j, \\ 0 & \text{otherwise.} \end{cases}$$

By default, $A_{i,i} = 0$. We choose the GraphLSTM model for the graph learning task. Here, the taxi demand-supply forecasting problem can be represented as learning a function $\mathcal{F}(\cdot)$ that maps the historical demand (or supply) to future predictions, given a graph depicting the Voronoi partitions:

$$\mathcal{F}([d_{1:t,1:N}]|\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{A})) = [d_{t+1:t+h,1:N}].$$

The models are compared using three error metrics, namely SMAPE (Symmetric Mean Absolute Percentage Error), MASE (Mean Absolute Scaled Error), and RMSE (Root Mean Square Error). These metrics are defined and discussed in Section IV. Fig. 2 highlights the flow of the investigation that will be conducted in this paper.

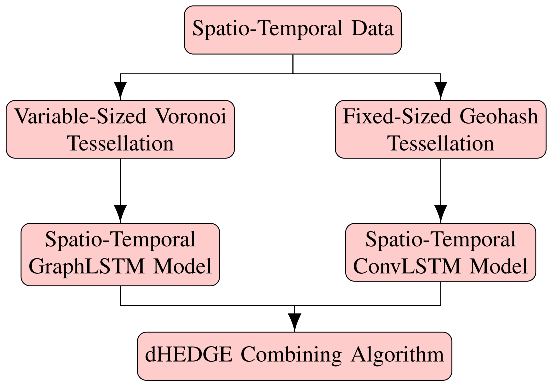

Spatio-Temporal Data
Variable-Sized Voronoi Tessellation
Fixed-Sized Geohash Tessellation
Spatio-Temporal GraphLSTM Model
Spatio-Temporal ConvLSTM Model
dHEDGE Combining Algorithm

Fig. 2. This is the flow chart of the study in this paper. The spatio-temporal data is first tessellated using the Voronoi and Geohash schemes. After modeling and comparing the tessellated data using convolutional and graphical LSTM models, ensemble learning is performed, which helps to achieve superior prediction performance.

TABLE I

Statistical Properties of the Taxi Demand-Supply Data Sets From Bengaluru and NYC, When Aggregated as Time-Sequences Over 60 Minutes Across January-February 2016

Statistical Properties	Bengaluru Demand	Bengaluru Supply	NYC Demand
Minimum	0	0	0
Maximum	630	2913	1582
Mean	14.1	10.1	19.5
Median	13	7	16
Skewness	0.7	5.8	1.1
Kurtosis	1.3	114.5	3.6
Standard Deviation	10.1	9.1	14.9
Periodicity (in time steps)	12, 24	12, 24, 168	12, 24, 168

III. Spatial Tessellation Schemes

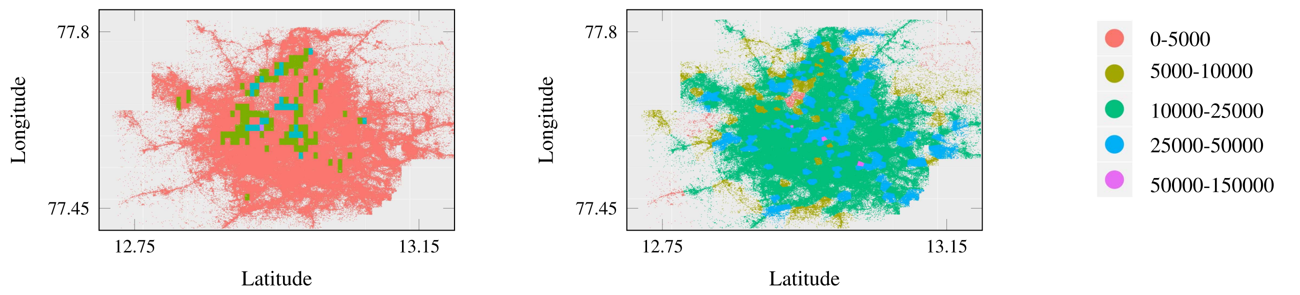
We consider three real-world data sets in our study; the taxi demand and driver supply data sets from the city of Bengaluru, India, and publicly available taxi demand data set from the city of New York, USA.

A. Description of the Data Sets

The Bengaluru taxi demand and driver supply data sets are acquired from a leading Indian e-hailing taxi service provider. The demand data contains GPS traces of taxi passengers booking a taxi by logging into their mobile application. The supply data contains GPS traces of fresh log-ins of taxi drivers, representing the available supply. The data sets are available for a period of two months, from January 1, 2016, to February 29, 2016. The data sets contain latitude-longitude coordinates of the passenger/driver, session duration, and the time stamps. The city of Bengaluru covers an area of approximately 740 km^2. The publicly available NYC (New York City) yellow taxi data set [31] contains GPS traces of street hailing yellow taxi services. For our study, we extract the pick-up locations and the time stamps from the period of January-February 2016. NYC is spread over an area of approximately 780 km^2. Some key statistical properties of the data sets are given in Table I.

B. Voronoi Tessellation

A prerequisite for the Voronoi tessellation is a set of generating sites that can be used to define the Voronoi cells. For this purpose, we use the K-Means clustering algorithm [32]. The algorithm has a linear memory and time complexity, which is ideal for our large data sets, and performs reasonably well in comparison with other clustering algorithms [33]. The data is classified into a predefined set of K clusters, and the centroids of these clusters act as generating sites for the Voronoi tessellation.



(a) Geohash tessellation (b) Voronoi tessellation (c) Demand density

Fig. 3. Heat maps obtained by partitioning the city of Bengaluru into 6-level geohashes and Voronoi cells. The spatial distribution of data in the partitions varies significantly with the tessellation scheme employed. While the Geohash scheme produces several data dense and scarce partitions, the Voronoi scheme uniformly distributes data across partitions.

The K-Means algorithm aims to minimize the squared error function J given as:

$$J = \sum_{j=1}^{K} \sum_{i=1}^{n} ||p_{(i)}^{j} - c_j||^2, \tag{1}$$

where, $||p_{(i)}^{j} - c_j||^2$ is the Euclidean distance between a data point $p_{(i)}^{j}$ and its center c_j, and n is the total number of data points. For efficient rerouting of vacant taxi drivers, we partition Bengaluru into 740 clusters and NYC into 780 clusters so that the average cluster area remains close to 1 km^2. Note that instead of applying a separate K-Means algorithm on the Bengaluru supply data set, we associate the supply data points with its nearest demand centroid. It enables us to do a comparative analysis of the demand and supply patterns associated with every demand region of interest. Then, the Centroidal Voronoi tessellation divides the space according to the nearest neighbor-rule, based on the K-Means centroids. Based on the closeness of centroids, this tessellation strategy produces polygon partitions of varying sizes, with a time complexity of O(nlogn).

C. Geohash Tessellation

Geohash tessellation is an extension of the basic grid partitioning technique with a naming convention. Each latitude-longitude coordinate is encoded into an alphanumeric string, where the length of the string denotes the *level* of the geohash. All latitude-longitude coordinates mapped to a specific string will be part of a unique fixed-sized rectangular grid. While a 5-level geohash spans an area of 4.9 km × 4.9 km, a 6-level geohash covers an area of 1.2 km × 0.6 km. For consistent comparison with Voronoi cells of average area 1 km^2, we employ 6 level-geohashes for our study. The time complexity of this algorithm is O(1).

The Voronoi and Geohash heat maps are plotted in Fig. 3. Based on a color scale that denotes the demand density in each cell, we see that the Geohash strategy produces a large fraction of demand scarce cells. The demand dense cells are concentrated around the city center. The Voronoi strategy, on the other hand, tends to uniformly distribute data and produces polygons of variable area.

IV. Spatio-Temporal Models

In the previous section, we saw that the spatial distribution of the data in each partition varies significantly with the partitioning technique employed (Fig. 3). In this section, we examine whether this variation in the spatial distribution has an impact on the performance of the prediction models employed in these partitions. Both the ConvLSTM and GraphLSTM models derive heavily from the LSTM network [34]. The RNN cell, from which the LSTM is developed, considers its present input and the output of the RNN cells preceding it, for its future output. The LSTM network overcomes several shortcomings of the plain RNN model to learn long-term temporal dependencies. This property makes it a suitable candidate for time-series analysis. An LSTM cell has four NN units, called *gates*, that interact with each other. Do note the equations below:

$$f_t = \delta(W_f \cdot X_t{}^l + R_f \cdot h_{t-1} + b_f), \tag{2}$$
$$i_t = \delta(W_i \cdot X_t{}^l + R_i \cdot h_{t-1} + b_i), \tag{3}$$
$$o_t = \delta(W_o \cdot X_t{}^l + R_o \cdot h_{t-1} + b_o), \tag{4}$$
$$\overline{C}_t = \varphi(W_c \cdot X_t{}^l + R_c \cdot h_{t-1} + b_c), \tag{5}$$
$$C_t = f_t \bullet C_{t-1} + i_t \bullet \overline{C}_t, \tag{6}$$
$$h_t = o_t \bullet \phi(C_t), \tag{7}$$

where, $X_t{}^l$ is the input and W_x, R_x, and b_x represent the input weights, recurrent weights, and bias of the gate x respectively. The forget gate, given by Eqn. (2), decides the amount of historical information to be discarded. The input gate in Eqn. (3) decides the values to be updated in the cell state, and the output gate outputs the cell states in Eqn. (4). The nonlinear activation functions δ, φ, and ϕ squish the outputs to recommended ranges, which are usually [0, 1] or [-1, 1]. The symbols $\cdot$ and $\bullet$ are used to denote matrix multiplication and element-wise product operations respectively. Eqn. (5) calculates a set of new candidate values to be added to the present cell state C_t. After the cell state is updated using Eqn. (6), the new hidden state output is given by Eqn. (7).

A. Geohash-Based ConvLSTM Model

The ConvLSTM model [16] combines the aspects of both LSTM and CNN models. The key equations are as follows:

$$f_t = \delta(W_f * X_t{}^c + R_f * h_{t-1} + b_f), \tag{8}$$
$$i_t = \delta(W_i * X_t{}^c + R_i * h_{t-1} + b_i), \tag{9}$$
$$o_t = \delta(W_o * X_t{}^c + R_o * h_{t-1} + b_o), \tag{10}$$
$$\overline{C}_t = \varphi(W_c * X_t{}^c + R_c * h_{t-1} + b_c), \tag{11}$$

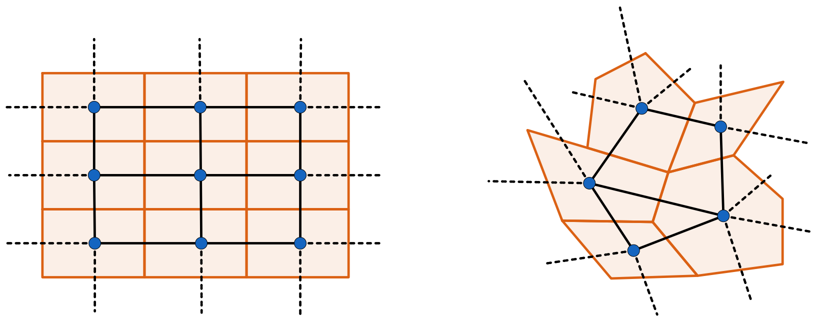


Fig. 4. Representation of the Geohash and Voronoi tessellations as graphs. For each graph, the nodes are the centroids of the partitions, and the edges are the distances between the centroids. These graphical representations facilitate the application of the GraphLSTM framework on the spatial partitions.

$$C_t = f_t \bullet C_{t-1} + i_t \bullet \overline{C}_t, \tag{12}$$
$$h_t = o_t \bullet \phi(C_t), \tag{13}$$

where, ${X_t}^c$ is the input and $*$ is the convolution operator. In mathematical terms, a ConvLSTM replaces the matrix multiplication operations in the feed-forward equations of the vanilla LSTM to convolution operations. If we consider the centroid of each partition as a node on a graph, the entire city can be represented by an undirected graph G. Each node represents a partition and will hold a value equal to the aggregated demand or supply for that partition. See Fig. 4 for cross-sections of the graphs obtained using the Voronoi and Geohash schemes. The Voronoi tessellated city will generate an irregular graph, where all the nodes need not have the same number of neighbors. We note that the number of neighbors varies from 3 to 10 for each Voronoi partition. On the other hand, the Geohash tessellated city forms a highly regular graph where each node has 8 neighbors, equidistant from each other. This fixed structure of a Geohash-based graph allows us to apply standard convolution operations and hence, can be modeled using a standard ConvLSTM technique. For an arbitrarily structured graph like the Voronoi-based graph, localized rectangular filter operations cannot be applied. A graph-based LSTM model that utilizes the adjacency matrix to depict the structure of a graph may be employed in such scenarios.

B. Voronoi-Based GraphLSTM Model

The primary step in a GraphLSTM framework [26] is to define the neighborhood. A k-hop neighborhood can be used to gather information from nodes that are k hops away from a node of interest. In this study, we gather spatial information from the first-order neighbors, *i.e.,* the neighbors who share a common boundary with the partition of interest. Hence, a 1-hop neighborhood is considered for the implementation of our GraphLSTM model. The 1-hop neighborhood matrix for any graph $\mathcal{G}$ is same as its adjacency matrix $\mathcal{A}$. To make the nodes self-accessible in the graph, the identity matrix I is added to $\mathcal{A}$, to form $\tilde{\mathcal{A}}$. Then, the 1-hop graph convolution at time t can be defined as follows:

$$\mathcal{GC}_t = (W_{gc} \bullet \tilde{\mathcal{A}}){X_t}^g \tag{14}$$

where, W_{gc} is the 1-hop weight matrix for the 1-hop adjacency matrix, and ${X_t}^g \in \mathbb{R}^{N\times 1}$ is the demand or supply at time t, where N is the number of Voronoi partitions. The features extracted from the graph convolution $\mathcal{GC}$ are fed to the LSTM network. We see that the structure of the gates, and the input cell state $\overline{C}_t$ at time t are similar to the vanilla LSTM. The input is replaced by the graph convolution features $\mathcal{GC}$. A new cell state ${C^*}_{t-1}$ to incorporate the contributions of neighboring cell states is added to the framework, where W_N is the corresponding weight matrix. The main equations are as follows:

$$f_t = \delta(W_f \cdot \mathcal{GC}_t + R_f \cdot h_{t-1} + b_f), \tag{15}$$
$$i_t = \delta(W_i \cdot \mathcal{GC}_t + R_i \cdot h_{t-1} + b_i), \tag{16}$$
$$o_t = \delta(W_o \cdot \mathcal{GC}_t + R_o \cdot h_{t-1} + b_o), \tag{17}$$
$$\overline{C}_t = \varphi(W_c \cdot \mathcal{GC}_t + R_c \cdot h_{t-1} + b_c), \tag{18}$$
$${C^*}_{t-1} = W_N \bullet \tilde{\mathcal{A}} \cdot C_{t-1}, \tag{19}$$
$$C_t = f_t \bullet {C^*}_{t-1} + i_t \bullet \overline{C}_t, \tag{20}$$
$$h_t = o_t \bullet \tanh(C_t). \tag{21}$$

With the addition of ${C^*}_{t-1}$ in Eqn. (20), the influence of the neighboring cell states will be considered during the recurrent updates of the cell state.

C. LSTM Model

LSTMs are state of the art NN models that are widely used in sequence learning applications [13]–[15]. Therefore, we compare the ConvLSTM and GraphLSTM networks with LSTM networks modeled using Voronoi and Geohash features. For a region r_i (Voronoi cell or geohash), we first feed the demand/supply from r_i alone to the LSTM network. Then, to analyze the effect of spatial neighbors, we feed data from r_i and its first-order neighbors to the LSTM network as input features. We vary the number of first-order neighbors to arrive at the best spatial configuration.

D. ARIMA and ARIMAX Models

We consider two models, ARIMA and ARIMAX, to explore the linear relationship between the data in adjacent partitions. Since these models have been traditionally employed for time-series prediction [8], they are suitable candidates for this comparison study. An ARIMA process can be represented as:

$$y_t = \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t, \tag{22}$$

TABLE II

THE OPTIMAL VALUES OBTAINED FROM THIS RANGE OF HYPER-PARAMETERS, AFTER PERFORMING THE TPE (TREE-STRUCTURED PARZEN ESTIMATOR) BASED BO (BAYESIAN OPTIMIZATION), ARE USED FOR MODELING THE TAXI DEMAND-SUPPLY DATA

Range of hyper-parameters that are fed to TPE-BO
Number of Layers = [1, 2]
Number of Neurons = [10, 20, 50, 100]
Dropout = Uniform (0,0.5)
Activation = [Sigmoid, Relu, Linear]
Optimizer = [Adam, Stochastic Gradient, RMSprop]
Learning Rate = $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]$
Batch Size = [64, 128]

where, y_t is the demand/supply at time t, p and ϕ are the order and parameters of the AR process, q and θ are the order and parameters of the MA process, and ε_t is the forecast error. The variable y_t may also be a differenced version of the demand/supply if non-stationarities exist in the original data set. Historical information from the variable of interest alone is taken into consideration for the standard ARIMA model. The ARIMAX model is an extension of ARIMA that provides a framework to include information from the neighboring regions (*i.e.,* covariates). We employ the ARIMAX to analyze the extent of spatial information captured with different tessellation schemes. The ARIMAX model is defined as:

$$\begin{aligned} y_t = \beta z_t + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} \\ + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t, \end{aligned} \tag{23}$$

where, z_t is the covariate at time t, and the parameter β includes the lagged versions of the covariate. The time-sequences from the positively correlated first-order neighboring regions serve as covariates in our study.

V. EXPERIMENTS

In this section, we discuss the hyper-parameter optimization techniques for the models, evaluation metrics, and inferences obtained on performing the comparison study.

A. Experimental Settings

Before modeling the data using any NN model, it is necessary to set the optimal hyper-parameters. Hyper-parameters are the model-specific properties that are to be fixed before the training phase of the model. They define the high-level properties of the model such as the time complexity or the learning rate. Out of the various algorithms available for hyper-parameter optimization, Bayesian Optimization is widely used in the recent machine learning literature [35]. In this study, we use the TPE-BO (Tree-structured Parzen Estimator-Bayesian Optimization) [36] approach for tuning the hyper-parameters. This algorithm uses Parzen estimators to model the error distribution as non-parametric densities. The range of the hyper-parameters fed to the TPE algorithm is given in Table II. The considered range covers the commonly adopted choices of various parameters. Additionally, for the ConvLSTM model, we vary the number of filters from 16 to 258. The shortlisted optimization function for our data sets is the RMSprop (Root Mean Square Propagation) [37] algorithm. The choice of the activation function at the output dense layer is Relu. The Relu activation is recommended for data sets such as passenger count and taxi supply as it allows the output to vary linearly from zero. The output is zero if the input to the activation function is less than zero. In case the input is greater than zero, the output is equal to the input. The dropout values, number of layers, and learning rates are optimized for each data set, tessellation technique, and NN model. In addition to the hyper-parameter values suggested by the TPE-BO, we manually tune the parameters to arrive at the best prediction accuracy.

The K-Means clustering algorithm generates a set of N regions of interest. The variable N takes values 740 and 780 for Bengaluru and NYC respectively. For each $n \in N$, two time-sequences are generated using the Voronoi and Geohash tessellation strategies. The data is aggregated over 60 minutes for 60 days, generating time-sequences of length $T = 1440$ time steps. To implement the GraphLSTM model for Voronoi tessellation, we pick the 1-hop neighbors for every node n. The model receives inputs of the form $X^g \in \mathbb{R}^{N \times T}$, along with an adjacency matrix $\tilde{A} \in \mathbb{R}^{N \times N}$. The adjacency matrix encapsulates information from the first-order neighbors. Here, we consider a hidden layer with dimension equal to the number of nodes in the graph. For the ConvLSTM model, we consider frames of size 3×3. This particular configuration allows us to capture information from a geohash of interest (the center pixel) and its 8 first-order neighbors. The ConvLSTM framework receives inputs of the form $X^c \in \mathbb{R}^{N \times T \times 3 \times 3 \times 1}$, from 3×3 frames along a single channel. For the LSTM network, the inputs are of the form $X^l \in \mathbb{R}^{N \times T \times S}$, where S is the number of spatial neighbors. Note that while a geohash has 8 fixed number of first-order neighbors, a Voronoi cell has a variable set of first-order neighbors. This corresponds to an S value of 8 for the Geohash LSTM model. For consistent comparison, while training the Voronoi LSTM model, we pick features from the top 8 positively correlated Voronoi neighbors. For Voronoi cells with less than 8 neighbors, we compensate for the lack of features by introducing invalid feature vectors to differentiate them from useful information.

For ARIMA and ARIMAX models, we vary the AR and MA parameters between the range [0, 5] and the time-sequences are differenced whenever non-stationary behavior is encountered. For both LSTM and ARIMAX models, we vary the number of spatial Voronoi and Geohash features included in the model, to find the best spatial configurations for each data set. All the NN models are trained with a batch size of 64 for 500 epochs and rerun 5 times to compensate for the random initialization of network weights. Min-Max scaling is applied to the inputs before they are fed to the various networks. An early stopping mechanism is employed to prevent over-fitting.[2]

[2]Codes are available at https://github.com/NDavisK/Grids-versus-Graphs

TABLE III

PREDICTION PERFORMANCE OF VARIOUS MODELS ACROSS DATA SETS WITH DIFFERENT PERFORMANCE METRICS, WHERE G AND V DENOTE GEOHASH AND VORONOI TESSELLATION SCHEMES RESPECTIVELY. THE NUMBERS IN BRACES ASSOCIATED WITH ARIMAX AND LSTM MODELS DENOTE THE NUMBER OF SPATIAL FEATURES THAT RESULTED IN THE BEST PERFORMANCE. FOR THE NN MODELS, THE ERRORS ARE GIVEN AS *Mean* $\pm$ *Standard Deviation*. THE GRAPHLSTM TECHNIQUE SHOWS AN OVERALL ROBUST PERFORMANCE, AND THE PROPOSED dHEDGE BASED PREDICTION TECHNIQUE ENSURES CONSISTENTLY GOOD PERFORMANCE ACROSS DIFFERENT SCENARIOS

Spatio-Temporal Models		**Bengaluru Demand**			**Bengaluru Supply**			**NYC Demand**		
		MASE	SMAPE	RMSE	MASE	SMAPE	RMSE	MASE	SMAPE	RMSE
ARIMA	G	1.31	48.5	12.36	18.0	191.1	57.3	1213.6	169.8	447.2
	V	1.13	43.5	8.14	4.68	95.4	15.8	5.01	125.5	27.6
ARIMAX	G	1.13 (1)	43.4 (1)	10.1 (1)	1.34 (8)	60.1 (8)	9.66 (8)	2.42 (4)	80.1 (2)	123.3 (2)
	V	1.20 (8)	47.2 (8)	8.93 (8)	1.31 (2)	58.7 (7)	8.03 (7)	1.84 (4)	94.0 (4)	15.6 (4)
LSTM	G	0.75 ± 0.26 (7)	16.14 ± 4.99 (7)	6.45 ± 4.51 (7)	0.93 ± 0.34 (6)	21.5 ± 5.65 (4)	7.11 ± 6.17 (4)	0.85 ± 0.41 (8)	23.2 ± 40.9 (4)	58.9 ± 34.5 (8)
	V	0.76 ± 0.16 (6)	16.72 ± 3.63 (6)	5.45 ± 2.42 (6)	0.98 ± 0.29 (2)	28.1 ± 30.8 (7)	6.68 ± 5.37 (2)	0.88 ± 0.19 (8)	29.3 ± 104 (0)	8.26 ± 5.73 (8)
ConvLSTM	G	0.37 ± 1.99	**9.1 ± 4.8**	**2.16 ± 1.52**	1.51 ± 2.23	35.2 ± 12.7	**7.73 ± 2.03**	40.5 ± 0.05	12.8 ± 15.4	36.8 ± 10.4
GraphLSTM	G	0.73 ± 0.20	15.6 ± 5.1	6.2 ± 4.75	**0.92 ± 0.38**	20.7 ± 5.9	6.79 ± 5.91	0.71 ± 0.68	**11.9 ± 4.56**	48.2 ± 29.3
	V	**0.72 ± 0.15**	16.1 ± 3.66	4.99 ± 2.35	0.93 ± 0.50	**21.8 ± 4.77**	6.15 ± 4.96	**0.68 ± 0.16**	17.4 ± 4.32	**6.33 ± 4.51**
dHEDGE		**0.32 ± 0.90**	**8.5 ± 3.7**	**2.25 ± 1.90**	**0.81 ± 0.96**	**17.7 ± 4.51**	**4.61 ± 2.73**	**0.43 ± 0.17**	**8.90 ± 3.6**	**6.48 ± 5.5**

B. Evaluation Metrics

For each data set, data from the first 59 days is used for training the models. This data is further divided into training and validation sets using a simple 90%-10% split. The models are then tested on the 60th day. We employ three widely used performance metrics to evaluate the models:

1) SMAPE (Symmetric Mean Absolute Percentage Error):

$$\mathrm{SMAPE} = \frac{100}{h}\sum_{t=1}^{h}\frac{|y_t - \hat{y}_t|}{\hat{y}_t + y_t + 1}, \tag{24}$$

2) MASE (Mean Absolute Scaled Error):

$$\mathrm{MASE} = \frac{1}{h}\sum_{t=1}^{h}\left(\frac{|y_t - \hat{y}_t|}{\frac{1}{n-m}\sum_{t=m+1}^{n}|y_t - y_{t-m}|}\right), \tag{25}$$

3) RMSE (Root Mean Square Error):

$$\mathrm{RMSE} = \sqrt{\frac{1}{h}\sum_{t=1}^{h}(y_t - \hat{y}_t)^2}, \tag{26}$$

where, h is the forecast horizon, m is the seasonal period, y_t is the actual demand, n is the length of the training set, and $\hat{y}_t$ is the forecast at time t. Both RMSE and SMAPE are scale dependent errors. The RMSE gives relatively high weights to large errors. The SMAPE is an accuracy measure based on percentage errors. The MASE compares the forecast errors of the test set with the in-sample forecast errors from the standard Naïve model, making it scale independent. Since these performance metrics are based on the L_1-norm and L_2-norm errors, we define the loss function over which the optimization is performed as:

$$\text{Loss function, L} = \frac{1}{N}\left(\sum_{t=1}^{N}(y_t - \hat{y}_t)^2 + |y_t - \hat{y}_t|\right). \tag{27}$$

Since the loss function contains squared and absolute errors, the models will optimize for both L_1 and L_2 errors.

C. Experimental Results

Table III summarizes the numerical results of our comparison study. The results are obtained by applying various models on the three data sets aggregated using Voronoi and Geohash tessellation schemes. The standard deviation factor accounts for the variability across different locations and multiple runs. The main inferences drawn from the investigation are as follows:

1) Even though the overall performance of linear regression models is sub-par with that of the non-linear neural models, the high computational speed and comparable performance in certain scenarios are to be noted.
2) The entire set of first-order neighbors may not be necessary to achieve the best spatial model configuration.
3) The prediction performance of the GraphLSTM method is better than that of the popular ConvLSTM method on the majority of test cases considered, and this was achieved with lower computational complexity.
4) Across data sets and metrics, the irregular Voronoi graph-based GraphLSTM model performs comparable to, or better than, the regular Geohash graph-based

GraphLSTM model. This highlights the potential of irregular graphs in location-based forecasting.

5) We note the lack of a universal winning tessellation strategy.

The ARIMA model achieves better prediction accuracy with Voronoi tessellation based input features than with Geohash features. The overall accuracy improved on incorporating spatial information through ARIMAX models. However, we notice that the ARIMAX models resulted in performance deterioration for the Bengaluru demand data set. To investigate this behavior further, we modeled the top 50 demand scarce and demand dense regions in Bengaluru using independent ARIMAX models, and saw clear improvements in accuracy. This observation points towards the inability of a single linear regression model to satisfactorily capture spatial correlations in the entire city. Hence, with regression models, we do not recommend modeling the entire city with a single model. Voronoi and Geohash based LSTM models show consistent improvements in accuracy on incorporating information from spatial neighbors. We also note that all the first-order neighbors might not be required to arrive at the best spatio-temporal model configuration.

The ConvLSTM model, based on the Geohash strategy, achieves good prediction accuracy on the Bengaluru data set but fails to perform well with the NYC data set. This trend is observed across different hidden layer and filter depth configurations. On the other hand, the Voronoi-based GraphLSTM model exhibits consistently high prediction performance across multiple scenarios for both cities. The poor performance of the Geohash-based ConvLSTM model with NYC demand data can be attributed to the highly skewed spatial data distribution. We note that 90% of the total data is concentrated around the Manhattan borough, leaving the other four boroughs with only 10% of the total demand. Employing a fixed-sized partitioning scheme in a non-uniform data distributed space is not efficient for modeling. Geohash partitioning results in a large number of demand scarce cells in some boroughs, affecting model performance. Meanwhile, K-Means based Voronoi tessellation attempts to uniformly distribute data in the partitions, resulting in a lower number of demand scarce cells. A GraphLSTM framework based on such an efficient setting achieves high prediction performance.

For further validation of this observation, we represent the Geohash partitions as nodes on a regular graph and conduct Geohash-based GraphLSTM modeling. We find that the Geohash-based GraphLSTM method is also unable to model the data satisfactorily, resulting in high variability in RMSE and MASE values. This highlights the importance of choosing an appropriate tessellation strategy, irrespective of the modeling technique used. In this case, the right choice of the partitioning technique resulted in an 80% improvement in the RMSE. Bengaluru demand-supply data sets have uniform spatial distributions, and hence, a Geohash-based model works fairly well. Therefore, we conclude that the choice of the spatial partitioning technique depends on the data distribution. Even then a Voronoi scheme based model exhibits competitive prediction performance at a lower computational cost. This shows that an appropriate tessellation strategy can reduce the complexity of the network to be built, without compromising on the prediction accuracy.

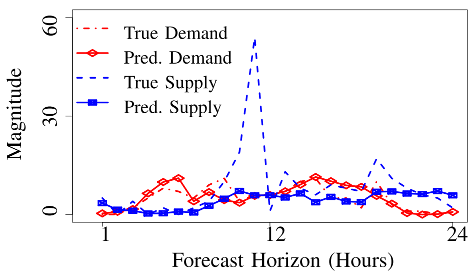


Fig. 5. The actual and predicted demand-supply patterns for a demand scarce Voronoi cell in Bengaluru. The supply forecast appears to be a better representation of the supply required in that region as opposed to the actual supply levels.

The GraphLSTM model has roughly the computational complexity of the vanilla LSTM, which is much lesser than that of the ConvLSTM model. While the ConvLSTM method has additional convolutional layers that increase the number of matrix operations, the GraphLSTM framework builds on the vanilla LSTM architecture, with an additional gate and some changes to the input. Note that we measure the computational complexity in terms of the matrix operations to be performed. The lower error variance of the GraphLSTM method in comparison with that of the ConvLSTM method suggests that the consistency in predictions is also maintained across various locations in the city. To summarize, the consistent performance of the GraphLSTM model, combined with low computational complexity, across scenarios in the context of location-based passenger demand and driver supply forecasting merits attention and needs further exploration.

While we stress on the significance of selecting an appropriate tessellation strategy while modeling, we observe that the best strategy and hence the prediction model, varies with the choice of data set and the chosen performance metrics. While the GraphLSTM model has an overall favorable performance across data sets, there are instances where the ConvLSTM model outperforms the GraphLSTM method (*e.g.*, in the Bengaluru demand data). For ensuring good prediction performance across all scenarios, we explore ensemble learning in the next section. Fig. 5 shows the demand-supply predictions from a GraphLSTM model in a Voronoi cell in Bengaluru. We see that the supply predicted from the historical data is a better match for the demand present in that region than the actual supply levels. Hence, a rerouting decision based on the predicted supply may reduce the demand-supply mismatch.

VI. Combining Models with a dHEDGE Algorithm

The applicability of ensemble learning in a non-stationary environment that involves tessellation strategies was put forward in [21], where we applied the dHEDGE ensemble learning algorithm to combine the tessellation strategies. We had observed that the regression models based on any one of the tessellation strategies were unable to yield optimal results for the entire forecast horizon. On exploring various LSTM networks, we note that this observation extends to the performance of the RNN models as well, thereby strengthening the observations in [21]. In Fig. 6, we see that the best

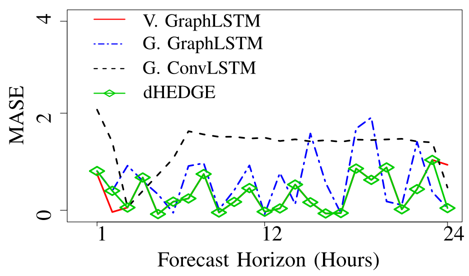


Fig. 6. Performance of the dHEDGE algorithm on the prediction models, where V. and G. denote the Voronoi and Geohash tessellation respectively. The algorithm closely follows the best strategy at each time step in the forecast horizon.

prediction technique varies with the time of the day. The best strategy switches between the Voronoi and Geohash tessellation based techniques throughout the forecast horizon. We find that irrespective of the models used, there is no universally superior tessellation strategy. This time dependency is in addition to the dependency of the strategies on the data set and performance metrics (see Table III).

Due to the lack of a consistent winning strategy, we employ a variant of the well-known HEDGE algorithm [38] suited for non-stationary environments known as the dHEDGE algorithm [30]. By exponentially reducing the weights associated with each expert (*i.e.,* the prediction model), the dHEDGE takes into account the non-stationarity of the process. In our case, we have three experts, Voronoi and Geohash based GraphLSTM models, and the ConvLSTM model. The weight initialization can be performed either uniformly, or based on some prior knowledge about the experts. We initialize the weights based on a holdout validation set. The weights are updated based on the previous weights, a discounting factor γ, a learning factor β, and a loss function l_t. The loss function l_t is based on the prediction errors observed by the experts at time t. Thus, for each time step t, the weights are updated for the i^{th} expert as:

$$w_i[t+1] = w_i[t]^{\gamma} \cdot \beta^{l_i[t]}. \tag{28}$$

The discounting and learning factors are chosen based on the validation set. The performance of the algorithm can be seen in Fig. 6. At each time step in the forecasting horizon, we pick the expert with the highest weight and use its predictions. We find that the algorithm picks the best shifting expert, by giving more weightage to the behavior of that expert in the recent past. The interested reader may refer to [21] and [30] for a detailed analysis of the algorithm. The prediction accuracy after applying the dHEDGE algorithm on the three experts can be seen in Table III. The algorithm consistently results in an accuracy that is close to the best expert in the pool.

To demonstrate the flexibility of our algorithm in adapting to various scenarios, we evaluate the performance of the algorithm on the NYC demand data set. While the Voronoi GraphLSTM model achieves good prediction accuracy, the two Geohash-based models perform poorly. When the dHEDGE is applied to these three experts, it is interesting to note that one can achieve accuracy close to that of the Voronoi GraphLSTM model, in spite of the poor performance by the other two models. Further, in some use cases, we note that combining the experts produces accuracy levels better than that of any of the individual experts. This finding is attributed to the time-dependent behavior of the models. In essence, the algorithm provides consistent performance across data sets and performance metrics, and aids in eliminating various dependencies associated with the prediction models.

VII. Concluding Remarks

In the context of e-hailing taxi services, generating accurate demand-supply forecasts is instrumental in minimizing customer waiting times and maximizing driver utilization. In this paper, we explored the impact of different spatial partitioning schemes on the predictive performance of LSTM (Long Short-Term Memory) models. NN (Neural Network) based taxi demand or supply forecasting commonly uses a fixed-sized equally-spaced spatial partitioning scheme. By comparing ConvLSTM (Convolutional LSTM) and GraphLSTM (Graph-based LSTM) models, we drew attention to the potential of learning the partitioned city structure as a graph and applying Graph-based NN models.

When evaluated on three large-scale real-world data sets, the GraphLSTM model emerged as a promising candidate for location-based taxi demand-supply forecasting. The GraphLSTM method offered competitive prediction performance against the ConvLSTM model at a much lower computational complexity. The comparison between GraphLSTM models based on regular and irregular graphs revealed the potential of irregular graphs in the context of location-based forecasting. In addition to the proposal to use an irregular graph-based GraphLSTM model for taxi demand-supply forecasting, the findings in this paper recommend the exploration and selection of a suitable tessellation strategy prior to fitting a NN model. The choice of a suitable prediction model was found to depend on the properties of the data set and the performance metrics employed.

To achieve superior performance across all scenarios, we recommend the dHEDGE based ensemble learning algorithm. By employing dHEDGE, in conjunction with the GraphLSTM and ConvLSTM models, we consistently achieved a prediction accuracy close to the best model at each time instant across the data sets considered, with different performance metrics.

A. Avenues for Further Research

This paper was directed towards the accurate forecasting of taxi demand and supply, where we highlighted the potential of Graph-based LSTM networks. A detailed analysis of the GraphLSTM should be performed using additional real-world data sets. Further, we note that demand-supply mismatches also occur when there are unexpected spikes in demand. In any future study, it is desirable to include such anomalous events in the modeling framework to achieve reliable predictions.

References

[1] K. L. Colborn *et al.*, “Spatio-temporal modelling of weekly malaria incidence in children under 5 for early epidemic detection in mozambique,” *Sci. Rep.*, vol. 8, no. 1, pp. 1–9, Dec. 2018.

[2] A. A. Ezzat, M. Jun, and Y. Ding, "Spatio-temporal asymmetry of local wind fields and its impact on short-term wind forecasting," *IEEE Trans. Sustain. Energy*, vol. 9, no. 3, pp. 1437–1447, Jul. 2018.

[3] X. Wang *et al.*, "Spatio-temporal analysis and prediction of cellular traffic in metropolis," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 2190–2202, Sep. 2019.

[4] C. Kamga, M. A. Yazici, and A. Singhal, "Analysis of taxi demand and supply in New York City: Implications of recent taxi regulations," *Transp. Planning Technol.*, vol. 38, no. 6, pp. 601–625, Aug. 2015.

[5] B. Jäger, M. Wittmann, and M. Lienkamp, "Analyzing and modeling a City's spatiotemporal taxi supply and demand: A case study for Munich," *J. Traffic Logistics Eng.*, vol. 4, pp. 147–153, Dec. 2016.

[6] Y. Tong *et al.*, "The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1653–1662.

[7] J. Ke, H. Zheng, H. Yang, and X. M. Chen, "Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach," *Transp. Res. C, Emerg. Technol.*, vol. 85, pp. 591–608, Dec. 2017.

[8] A. M. Nagy and V. Simon, "Survey on traffic prediction in smart cities," *Pervas. Mobile Comput.*, vol. 50, pp. 148–163, Oct. 2018.

[9] H. Nguyen, L.-M. Kieu, T. Wen, and C. Cai, "Deep learning methods in transportation domain: A review," *IET Intell. Transp. Syst.*, vol. 12, no. 9, pp. 998–1004, Nov. 2018.

[10] Y. Wang, D. Zhang, Y. Liu, B. Dai, and L. H. Lee, "Enhancing transportation systems via deep learning: A survey," *Transp. Res. C, Emerg. Technol.*, vol. 99, pp. 144–163, Feb. 2019.

[11] G. Petneházi, "Recurrent neural networks for time series forecasting," 2019, *arXiv:1901.00069*. [Online]. Available: http://arxiv.org/abs/1901.00069

[12] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Nov. 2016, pp. 324–328.

[13] C.-C. Chiu *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 4774–4778.

[14] D. Yogatama, C. Dyer, W. Ling, and P. Blunsom, "Generative and discriminative text classification with recurrent neural networks," 2017, *arXiv:1703.01898*. [Online]. Available: http://arxiv.org/abs/1703.01898

[15] H. Rutagemwa, A. Ghasemi, and S. Liu, "Dynamic spectrum assignment for land mobile radio with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.

[16] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 802–810.

[17] H. Yao *et al.*, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2588–2595.

[18] S. Liao, L. Zhou, X. Di, B. Yuan, and J. Xiong, "Large-scale short-term urban taxi demand forecasting using deep learning," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2018, pp. 428–433.

[19] D. Wang, W. Cao, J. Li, and J. Ye, "DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 243–254.

[20] D. Wang, Y. Yang, and S. Ning, "DeepSTCL: A deep spatio-temporal ConvLSTM for travel demand prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[21] N. Davis, G. Raina, and K. Jagannathan, "Taxi demand forecasting: A HEDGE-based tessellation strategy for improved accuracy," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3686–3697, Nov. 2018.

[22] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, vol. 501. Hoboken, NJ, USA: Wiley, 2009.

[23] G. Neimeyer. (2013). *Geohash Tips & Tricks*. Accessed: Apr. 15, 2020. [Online]. Available: http://geohash.org/site/tips.html

[24] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 24, 2020, doi: 10.1109/TNNLS.2020.2978386.

[25] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–16.

[26] Z. Cui, K. Henrickson, R. Ke, Z. Pu, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," 2018, *arXiv:1802.07007*. [Online]. Available: http://arxiv.org/abs/1802.07007

[27] K. Chen *et al.*, "Dynamic spatio-temporal graph-based CNNs for traffic prediction," 2018, *arXiv:1812.02019*. [Online]. Available: http://arxiv.org/abs/1812.02019

[28] X. Geng *et al.*, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 3656–3663.

[29] N. Davis, G. Raina, and K. Jagannathan, "Taxi demand-supply forecasting: Impact of spatial partitioning on the performance of neural networks," in *Proc. NIPS Workshop Mach. Learn. Intell. Transp. Syst.*, 2018, pp. 1–7.

[30] V. Raj and S. Kalyani, "An aggregating strategy for shifting experts in discrete sequence prediction," 2017, *arXiv:1708.01744*. [Online]. Available: http://arxiv.org/abs/1708.01744

[31] N. Y. C. Taxi & Limousine Commission. (2016). *TLC Trip Record Data*. Accessed: Jan. 15, 2020. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[32] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symp. Math. Statist. Probab.*, 1967, pp. 281–297.

[33] B. Chaudhari and M. Parikh, "A comparative study of clustering algorithms using WEKA tools," *Int. J. Appl. Innov. Eng. Manage.*, vol. 1, pp. 154–158, Oct. 2012.

[34] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, pp. 2451–2471, Oct. 2000.

[35] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.

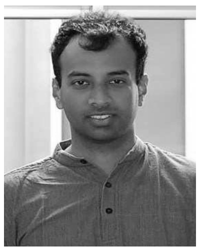
[36] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2546–2554.

[37] T. Tieleman and G. Hinton, "Rmsprop: Divide the gradient by a running average of its recent magnitude. Coursera," in *Proc. Neural Netw. Mach. Learn.*, 2012, p. 31.

[38] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Proc. Eur. Conf. Comput. Learn.*, 1995, pp. 23–37.

Neema Davis received the B.Tech. degree in electronics and communication engineering from Mahatma Gandhi University, Kottayam, in 2013. She is currently pursuing the Dual M.S. and Ph.D. degrees in electrical engineering from the Indian Institute of Technology Madras. Her research interests include transportation planning, predictive data analysis, and intelligent transportation systems.

Gaurav Raina received the Ph.D. degree in mathematics from the Trinity College, University of Cambridge, U.K. He is currently a Faculty Member in electrical engineering with IIT Madras, and also a Visiting Professor of mathematics and computer science with Krea University. He also serves on the Academic Council of Krea University, and also the Chairman of the Mobile Payment Forum of India. His research interests span the design and control of large scale systems, like the Internet and transportation networks.



Krishna Jagannathan received the B.Tech. degree in electrical engineering from IIT Madras and the S.M. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT). He is currently an Associate Professor with the Department of Electrical Engineering, IIT Madras. His research interests include the stochastic modeling and analysis of communication networks, transportation networks, network control, and queuing theory.