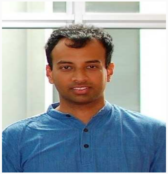
Demand Modeling in Modern Taxi Services

Making Data Science Work in the Real-World

Neema Davis

Gaurav Raina

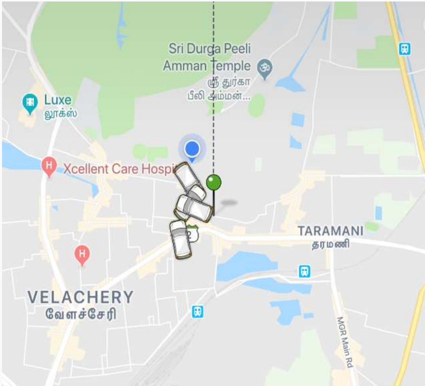


Krishna Jagannathan

Outline

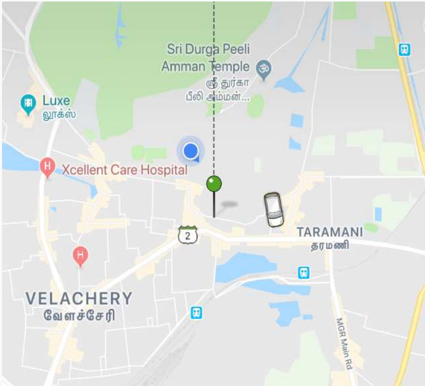
Real-World Connection

Understanding the Problem

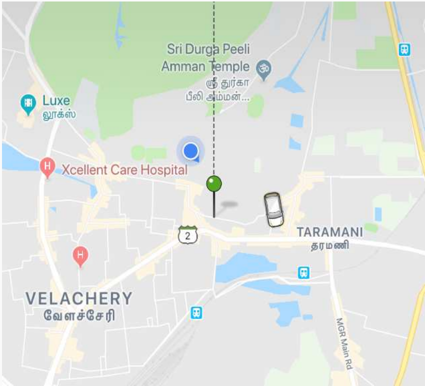


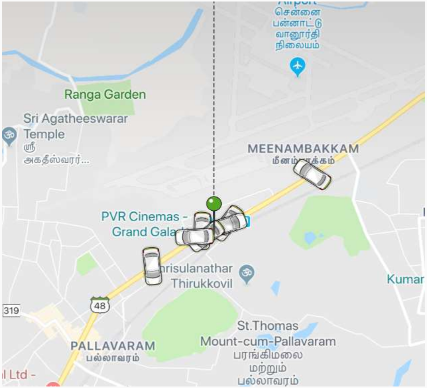
FARE
1X Base Price

Understanding the Problem



FARE
1.5X Base Price

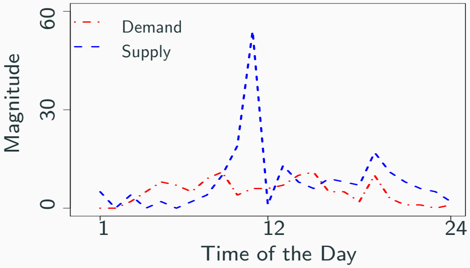




Demand-Supply Mismatch!

Crucial to develop accurate location-based demand forecasts

Understanding the Problem



Actual demand-supply patterns near Bengaluru city center

Understanding the Problem

What do we want?

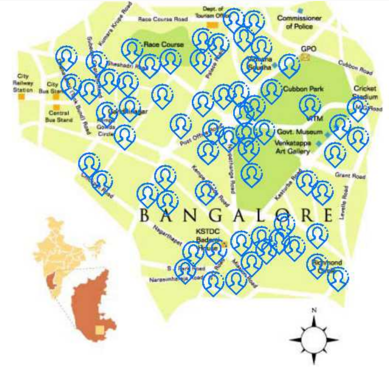
Accurate modeling of taxi demand

Why is this important?

Demand - Supply imbalance in online taxi hailing services

- Scarcity of taxis in peak hours
- Under-utilized taxis in off-peak hours

Understanding the Problem



Lots and lots of data!

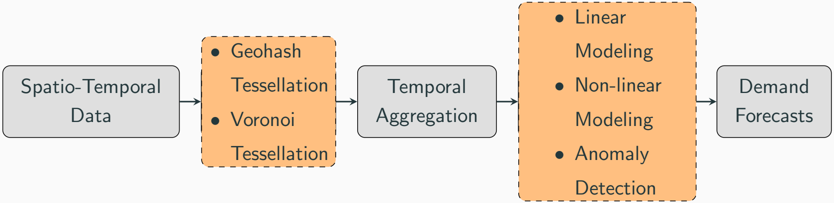
Where do we start?

Conceptual Framework

Location-Based Prediction Framework

Location-Based Prediction Framework

Break the problem into blocks



- By analyzing intermediate blocks, prediction accuracy can be improved
- Explore **spatial tessellation strategies** and **spatio-temporal modeling techniques**

Spatio-Temporal Data

Spatio-Temporal Data → Spatial Tessellation → Temporal Aggregation → Spatio-Temporal Modeling → Demand Forecasts

Bengaluru

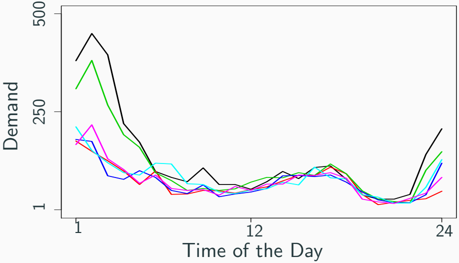
- GPS traces of taxi passengers booking a taxi by logging into the mobile app
- 15 million data points

New York[1]

- GPS traces of government-run street hailing Yellow taxis
- Differs from Bengaluru data set both in terms of data volume and city structure
- 21 million data points

[1] http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

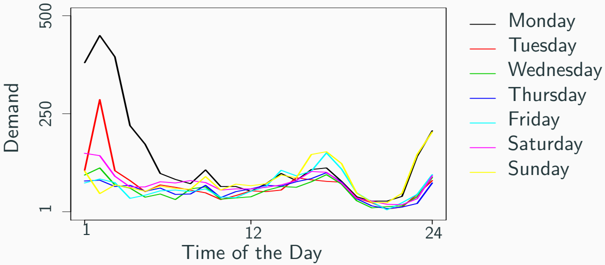
Spatio-Temporal Data



Demand patterns over consecutive Mondays near Bengaluru city center

- Demand shows **temporal** correlations

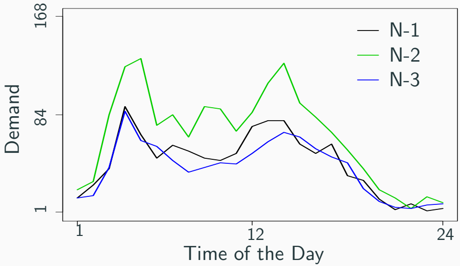
Spatio-Temporal Data



Demand patterns over days of a week near Bengaluru city center

- Demand shows **temporal** correlations

Spatio-Temporal Data



Demand patterns in spatial neighbors near Bengaluru city center
(N denotes Neighbor)

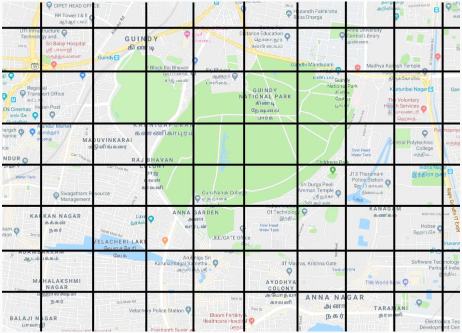
- Demand shows **temporal** correlations
- Demand shows **spatial** correlations

We need models that can capture both correlations

Spatial Partitioning Techniques

Spatial Partitioning

Spatial Partitioning



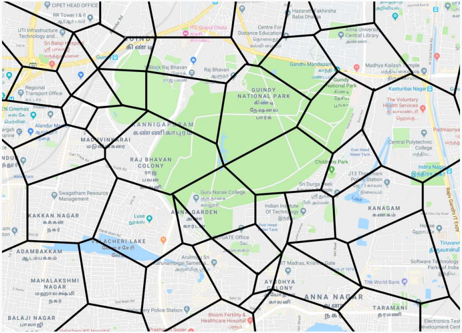
City commonly divided into fixed-sized equidistant grids

Pros?

Easy!

Cons?

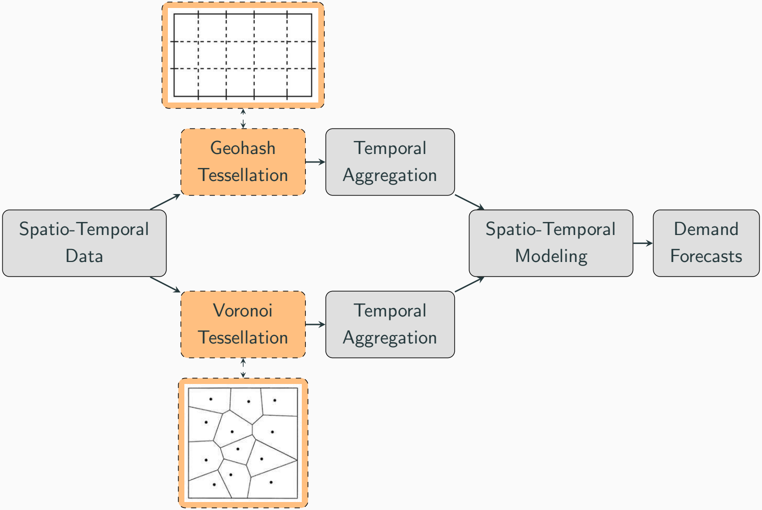
Data scarce cells



How about dividing the city into variable-sized grids?

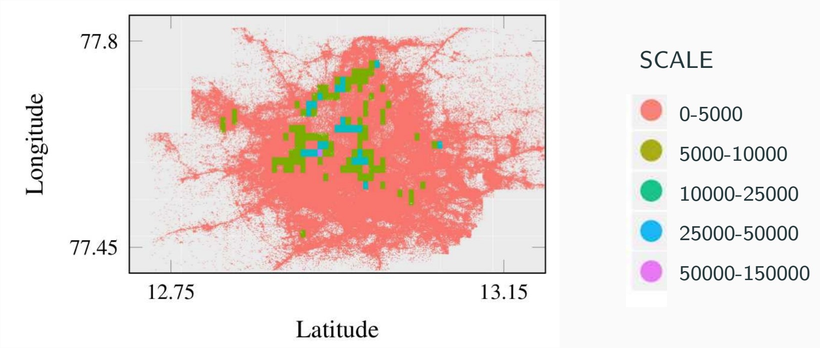
Partitioning based on the demand data distribution

Spatial Partitioning



Geohash tessellation

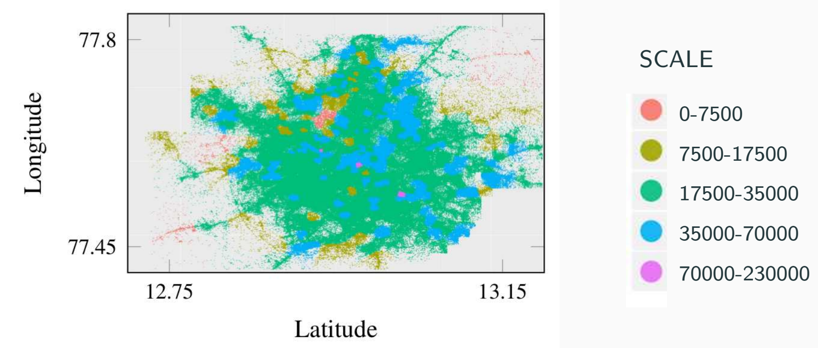
- Encodes any coordinate into alphanumeric string
- Higher the level, longer the string, higher the precision
- A 6 level geohash covers a rectangular area of 1.2×0.6 km



Heat map of Geohash tessellated Bengaluru

Voronoi tessellation

- Requires K-Means Clustering
- Classify data into K clusters, where K $\sim$ area of the city
- Divides space based on closeness of cluster centroids



Heat map of Voronoi tessellated Bengaluru

Spatio-Temporal Models

Spatio-Temporal Modeling

Spatio-Temporal Data → Spatial Tessellation → Temporal Aggregation → Spatio-Temporal Modeling → Demand Forecasts

Common Modeling Techniques

$\text{Demand}_t = f(\text{Demand}_{1:t-1})$

- $f = \text{Linear}$[2] → ARIMA, STL, TBATS
- $f = \text{Non-linear}$[3] → RNN, LSTM, ConvLSTM

[2] A. M. Nagy et al, "Survey on traffic prediction in smart cities," *Pervasive and Mobile Computing*, vol. 50, pp. 148163, 2018.

[3] G. Petneházi, "Recurrent neural networks for time series forecasting," *arXiv preprint arXiv:1901.00069*, 2019.

Linear Models: Regression

Auto Regressive Moving Average (ARMA)

$$\hat{y}_{t+h|t} = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + e_t$$

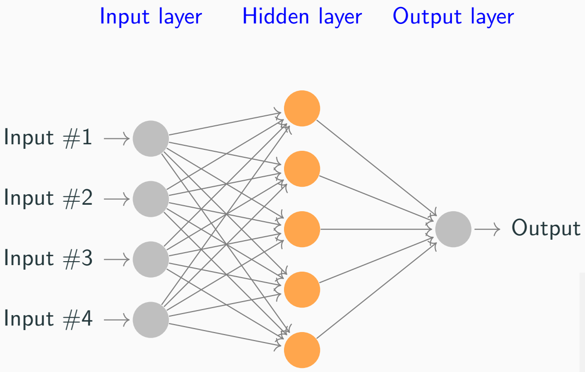
ϕ	AR parameter
θ	MA parameter
e_{t-i}	Forecast error at $t - i$

Seasonal and Trend decomposition with Loess (STL)

$$y_t = s_t + A_t$$

Non Seasonal s_t modeled using Exponential Smoothing or ARMA, and Seasonal A_t modeled using Seasonal Naive model

Non-Linear Models: Neural Networks



Forward Pass

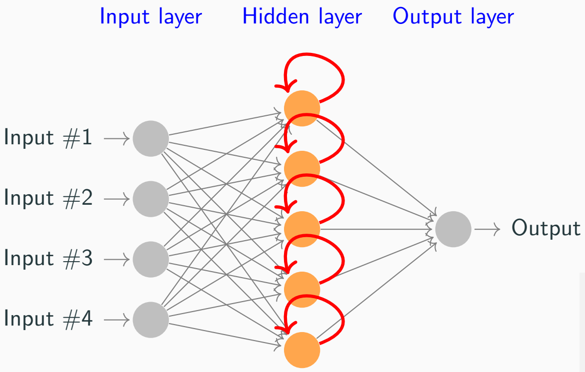
- Feed Information
- Calculate Loss, $L(w)$

Backward Pass

- Aim: Reduce Loss
- Calculate Gradients, $\frac{\partial L}{\partial w}$
- Update Weights

Standard Feed-Forward Neural Networks

Non-Linear Models: Neural Networks



Forward Pass

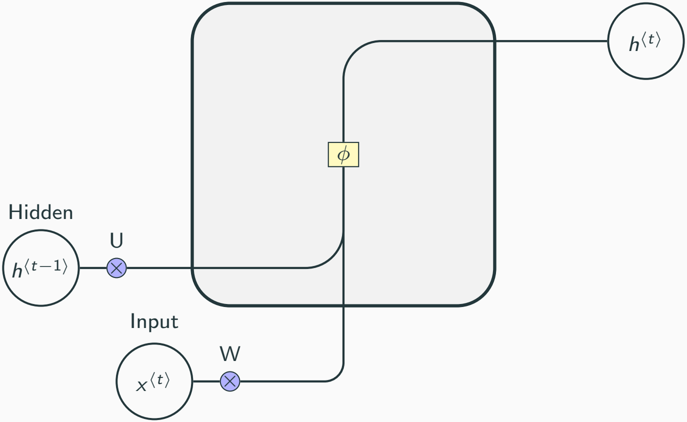
- Feed Information
- Calculate Loss, $L(w)$

Backward Pass

- Aim: Reduce Loss
- Calculate Gradients, $\frac{\partial L}{\partial w}$
- Update Weights

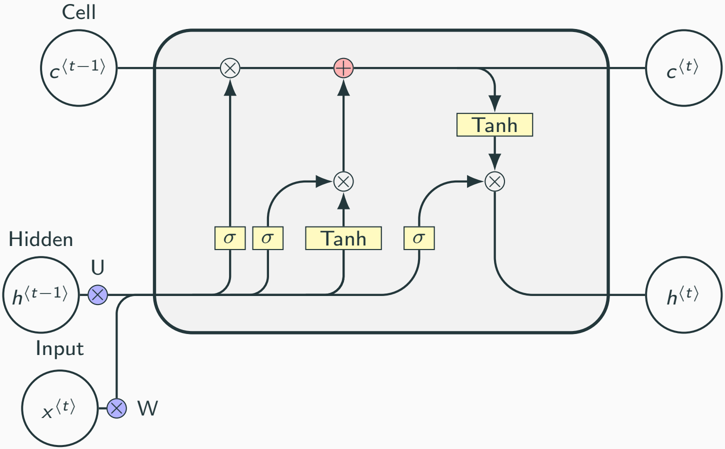
Recurrent Neural Networks (RNNs)

Non-Linear Models: Neural Networks



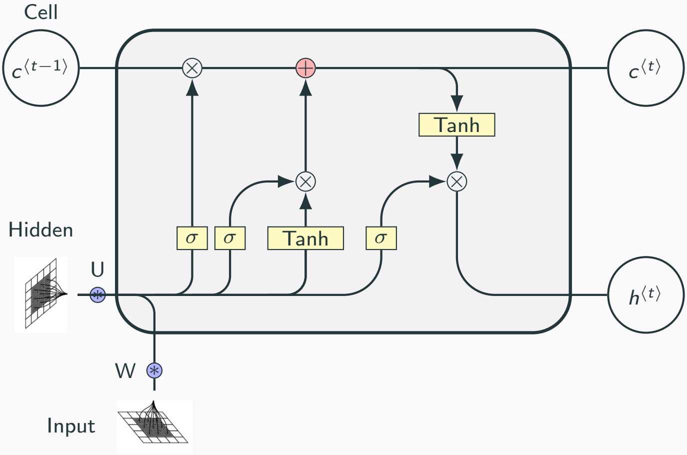
RNN unit: No memory!

Non-Linear Models: Neural Networks



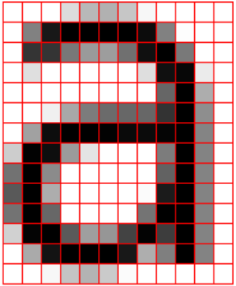
LSTM unit: Remembers the past, but no spatial memory!

Non-Linear Models: Neural Networks



Convolutional LSTM unit: Captures temporal and spatial information!

Image



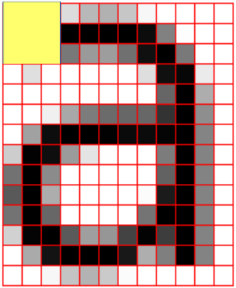
1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

Filter

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

Convolutional Neural Networks (CNNs)

Image

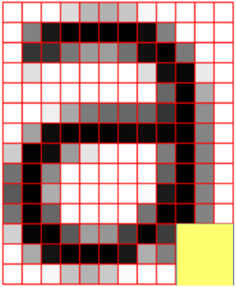


			0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
			0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
			0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

$$
\begin{aligned}
b_{11} = {} & w_{11}a_{11} + w_{12}a_{12} \\
& + w_{13}a_{13} + w_{21}a_{21} \\
& + w_{22}a_{22} + w_{23}a_{23} \\
& + w_{31}a_{31} + w_{32}a_{32} \\
& + w_{33}a_{33}
\end{aligned}
$$

Convolutional Neural Networks (CNNs)

Image



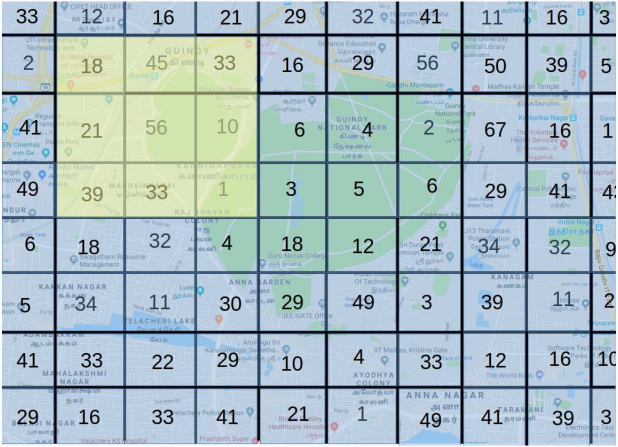
1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5			
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8			
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0			

$$
\begin{aligned}
b_{ij} = {} & w_{11} a_{ij} + w_{12} a_{i(j+1)} \\
& + w_{13} a_{i(j+2)} + w_{21} a_{(i+1)j} \\
& + w_{22} a_{(i+1)(j+1)} + w_{23} a_{(i+1)(j+2)} \\
& + w_{31} a_{(i+2)j} + w_{32} a_{(i+2)(j+1)} \\
& + w_{33} a_{(i+2)(j+2)}
\end{aligned}
$$

Is ConvLSTM Enough?

Geohash partitioned city

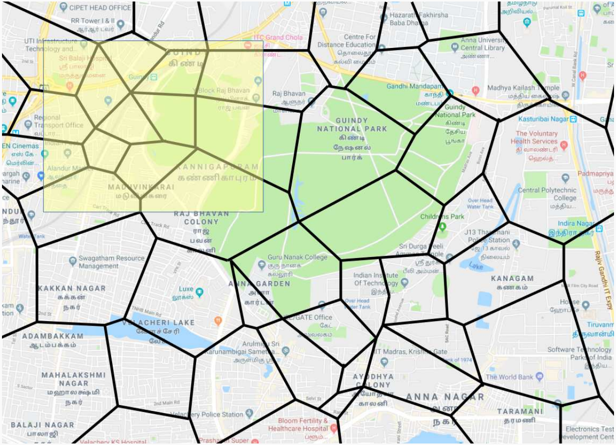
ConvLSTM is suitable



Is ConvLSTM Enough?

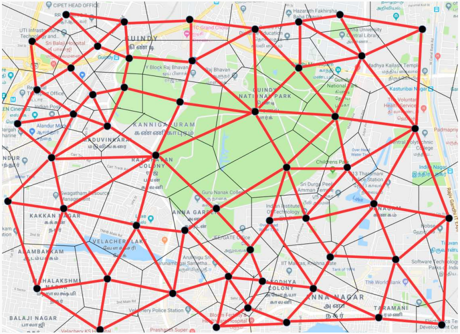
Voronoi partitioned city

ConvLSTM fails!



How Do We Make Progress?

- Standard convolutional filters cannot be applied on variable sized Voronoi partitions
- But, they can be represented as nodes on a graph



LSTM for Graphs

- Can an LSTM receive inputs from a graph? – Let's explore
- Create an Adjacency matrix

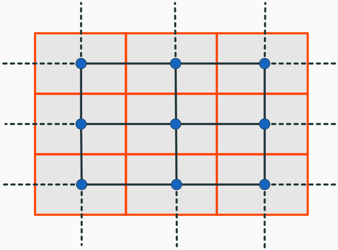
Consider 5 nodes; node 1 connected to nodes 2,3, node 2 connected to node 1, $\cdots$

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ & & \cdots\cdots & & \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

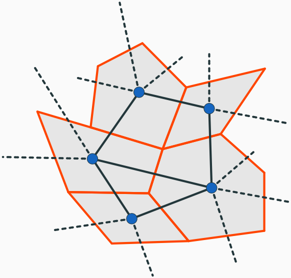
- Consider the operation: $(W_{gc} \bullet \mathcal{A})X_t$
- GraphLSTM = LSTM with inputs of the above form[4]

[4] Z. Cui et al, "High-order graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *arXiv preprint arXiv:1802.07007*, 2018.

Where is GraphLSTM Applicable?



Geohash tessellation



Voronoi tessellation

GraphLSTM is applicable to diverse partitioning schemes!

Clear advantage over ConvLSTM

Summary of Models

LSTM:

$$f_t = \delta(W_f \cdot X_t + U_f \cdot h_{t-1} + b_f)$$ Forget Gate

$$i_t = \delta(W_i \cdot X_t + U_i \cdot h_{t-1} + b_i)$$ Input Gate

$$o_t = \delta(W_o \cdot X_t + U_o \cdot h_{t-1} + b_o)$$ Output Gate

$$\overline{C}_t = \varphi(W_c \cdot X_t + U_c \cdot h_{t-1} + b_c)$$ Cell State

$$C_t = f_t \bullet C_{t-1} + i_t \bullet \overline{C}_t$$ Cell State Update

$$h_t = o_t \bullet \phi(C_t)$$ Hidden State Output

where,

X_t = input at t

W_x, U_x, & b_x = weights & bias associated with gate x

$\cdot$ = matrix multiplication

$\bullet$ = element-wise product operation

ConvLSTM (for Geohashes):

$$f_t = \delta(W_f * X_t + U_f * h_{t-1} + b_f)$$ Forget Gate

$$i_t = \delta(W_i * X_t + U_i * h_{t-1} + b_i)$$ Input Gate

$$o_t = \delta(W_o * X_t + U_o * h_{t-1} + b_o)$$ Output Gate

$$\overline{C}_t = \varphi(W_c * X_t + U_c * h_{t-1} + b_c)$$ Cell State

$$C_t = f_t \bullet C_{t-1} + i_t \bullet \overline{C}_t$$ Cell State Update

$$h_t = o_t \bullet \phi(C_t)$$ Hidden State Output

where,

$*$ = convolution operator

GraphLSTM (for all kinds of partitions):

$f_t = \delta(W_f \cdot \boxed{\mathcal{GC}_t} + U_f \cdot h_{t-1} + b_f))$	Forget Gate
$i_t = \delta(W_i \cdot \boxed{\mathcal{GC}_t} + U_i \cdot h_{t-1} + b_i)$	Input Gate
$o_t = \delta(W_o \cdot \boxed{\mathcal{GC}_t} + U_o \cdot h_{t-1} + b_o)$	Output Gate
$\overline{C}_t = \varphi(W_c \cdot \boxed{\mathcal{GC}_t} + U_c \cdot h_{t-1} + b_c)$	Cell State
$\boxed{C^*{}_{t-1} = W_N \bullet \mathcal{A} \cdot C_{t-1}}$	Spatial Information
$C_t = f_t \bullet \boxed{C^*{}_{t-1}} + i_t \bullet \overline{C}_t$	Cell State Update
$h_t = o_t \bullet \tanh(C_t)$	Hidden State Output

where,

$\mathcal{GC}_t = (W_{gc} \bullet \mathcal{A})X_t$: input at t

Evaluation Metrics

$$\textbf{Root Mean Square Error} = \sqrt{\frac{1}{h}\sum_{t=1}^{h}(y_t - \hat{y}_t)^2}$$

$$\textbf{Symmetric Mean Absolute Percent Error} = \frac{100}{h}\sum_{t=1}^{h}\frac{|y_t - \hat{y}_t|}{\hat{y}_t + y_t}$$

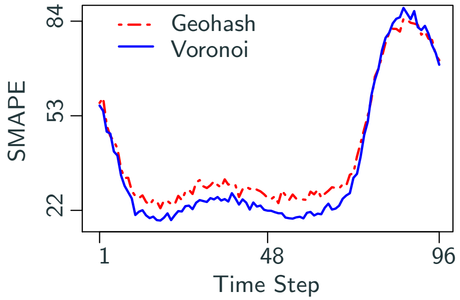
$$\textbf{Mean Absolute Scaled Error} = \frac{1}{h}\sum_{t=1}^{h}\left(\frac{|y_t - \hat{y}_t|}{\frac{1}{N-m}\sum_{t=m+1}^{N}|y_t - y_{t-m}|}\right)$$

N	Number of samples
y_t, $\hat{y}_t$	Actual, Predicted
m	Seasonal period
h	Forecast horizon

Observations

- Non-linear deep learning models show better prediction performance than linear regression models
- GraphLSTM has competitive prediction performance against ConvLSTM at lower computational complexity
- Models show non-stationary behavior

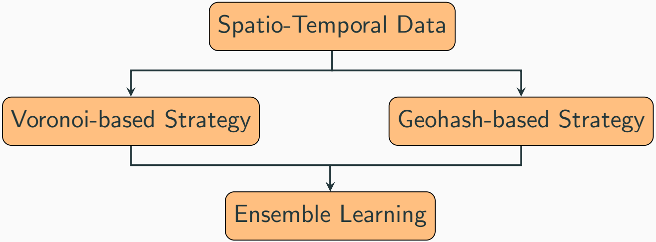
Non-Stationary Behavior of Models



Overall winner need not mean winner at all time steps

Need superior performance at each time step in horizon

Ensemble Learning



Modified dHEDGE Algorithm

Algorithm: Modified dHEDGE for choosing the best expert

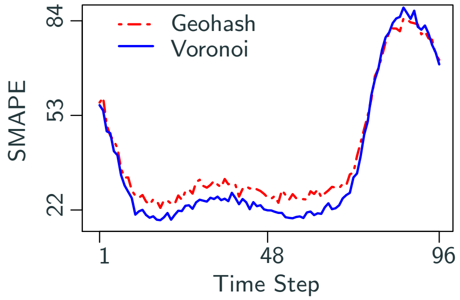
Parameters: Learning parameter $\beta \in [0, 1]$, Discounting factor $\gamma \in [0, 1]$

Initialization: Set $w_i[1] = W > 0$ for $i \in 1, 2$ s.t $\sum_{i=1}^{2} w_i[1] = 1$

for $t = 1, 2, \ldots$ **do**

- Choose expert with index $= \underset{i}{\operatorname{argmax}}(w_i[t])$
- Error e_v = MEAN(forecast errors from models based on Voronoi at t)
- Error e_g = MEAN(forecast errors from models based on Geohash at t)
- Loss L at t, $l_1[t] = \frac{e_v}{e_v + e_g}$, $l_2[t] = \frac{e_g}{e_v + e_g}$
- Update weights as $w_i[t+1] = w_i[t]^{\gamma} \cdot \beta^{l_i[t]}$

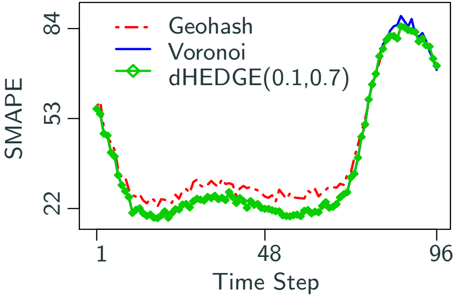
Modified dHEDGE Algorithm



Errors from a 24 hour ahead forecast

Let's apply the algorithm on the prediction models

Modified dHEDGE Algorithm

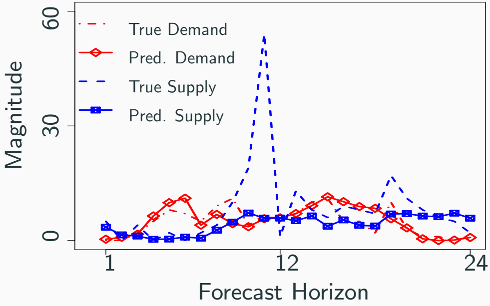


Errors from a 24 hour ahead forecast

Our algorithm tries to pick the best model at each instant

Towards Demand-Supply Equilibrium

Billion dollar question: Can accurate demand forecasting help achieve equilibrium? Yes!



Actual vs predicted demand-supply patterns near Bengaluru city center

The Work so Far

Spatio-Temporal Data → Spatial Aggregation → Temporal Aggregation → Spatio-Temporal Modeling → Demand Forecasts

- Explored fixed and variable spatial partitioning
- No universal winner
- Proposed modified dHEDGE

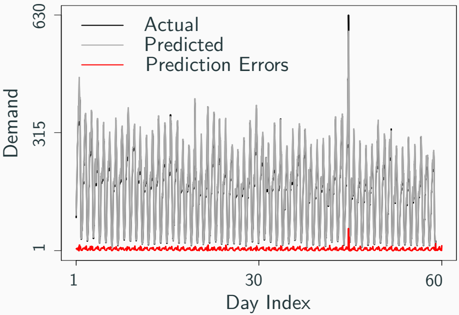
- Explored various time-series models
- Exploited spatial information
- LSTM variants better than regression models
- Potential of GraphLSTM

Abnormal Data & Anomaly Detection

Need for Such Analysis

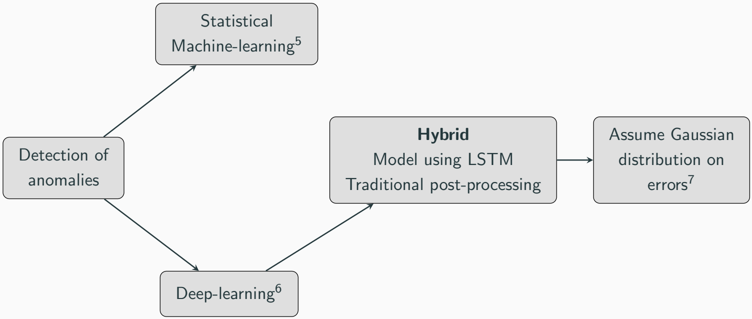
Anomaly Detection

Why? Forecasting models can learn *only* periodic patterns



Demand patterns over two months near Bengaluru city center

Anomaly Detection: Prior Art

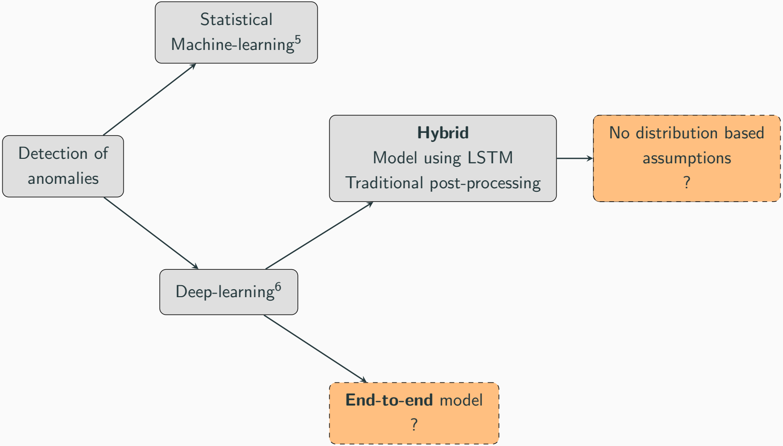


[5] V. Chandola et al, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, pp. 15–63, 2009.

[6] C. Raghavendra et al, "Deep-Learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[7] P. Malhotra et al. "Long short term memory networks for anomaly detection in time series,"" *ESANN*, 2015.

Anomaly Detection: Prior Art



[5] V. Chandola et al, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, pp. 15–63, 2009.

[6] C. Raghavendra et al, "Deep-Learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[7] P. Malhotra et al. "Long short term memory networks for anomaly detection in time series,"" *ESANN*, 2015.

Anomaly Detection: Contributions

- **Hybrid** deep anomaly detection
  - Proposed detection strategy based on Extreme Value Theory
  - "Extreme values of any distribution follow a Generalized Pareto Distribution regardless of the parent distribution"
  - Extreme values = high prediction errors $\rightarrow$ possible anomalies
- **End-to-end** deep anomaly detection
  - Proposed end-to-end EVT-LSTM model
  - Networks not customized for detection in hybrid models
  - Modify objective function to detect anomalies directly

Hybrid Deep Anomaly Detection

Common Practice

Assume Gaussian distribution on the prediction errors

- Computationally efficient
- Mathematically tractable
- Does not require large memory storage (unlike clustering)
- No kernel functions (unlike Support Vector Machines)
- Only if assumptions regarding underlying data distribution hold, provides a statistically justifiable solution for anomaly detection

Devise a strategy with all the advantages, but without this limitation?

Extreme Value Theory (EVT)

Extreme values of any distribution follow a Generalized Pareto Distribution (GPD) regardless of the parent distribution

- High prediction error

  $\rightarrow$ an instance the model has not seen before

  $\rightarrow$ possible anomaly

- Anomalous instances lie on distribution tail of prediction errors
- EVT can be applied!
  - all advantages of distribution based detection without making critical assumptions about actual distribution

EVT-Based Detection Strategy

Random Variable X with CDF P($X \leq x$)
Tail P($X > x$)

Set initial threshold at 98% quantile t

Excesses X-t follows $G_\gamma : x \to \exp\left(- \left(1 + \frac{\gamma}{\sigma} x\right)^{-\frac{1}{\gamma}}\right),\ \gamma \in \mathbb{R},\ 1 + \frac{\gamma}{\sigma} x > 0$

Estimate parameters using MLE

Calculate τ_e on an initial stream

$$\tau_e = t + \frac{\hat{\sigma}}{\hat{\gamma}}\left(\left(\frac{qn}{N_t}\right)^{-\hat{\gamma}} - 1\right)$$

Dynamic updation

$P(X > \tau_e) < q$? Anomaly

A. Siffer et al, "Anomaly detection in streams with extreme value theory," in *International Conference on Knowledge Discovery and Data Mining*, 2017.

Baseline Detection Strategies

Gaussian

Fit train prediction errors to $\mathcal{N}(\mu, \sigma^2)$ using MLE

↓

Set manual threshold τ_g based on validation prediction errors

↓

Check test set based on τ_g

Assumes parent distribution

Tukey

Calculate first and third quantiles Q_1, Q_3 from all prediction errors

↓

Calculate threshold $\tau_t = Q_3 + 3 \times (Q_3 - Q_1)$

↓

Find anomalies based on τ_t

Assumes no distribution

Evaluation Metrics

$$\textbf{Precision},\ P = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\textbf{Recall},\ R = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\textbf{F-score},\ F = 2 \cdot \frac{P \times R}{P + R}$$

where,

TP: Correctly predicted anomalies

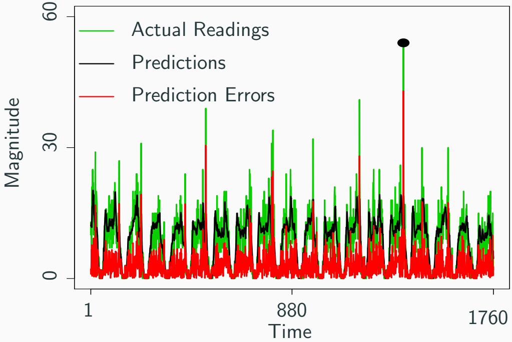
FP: Non-anomalies incorrectly identified as anomalies

FN: Anomalies incorrectly identified as non-anomalies

Experiments and Observations

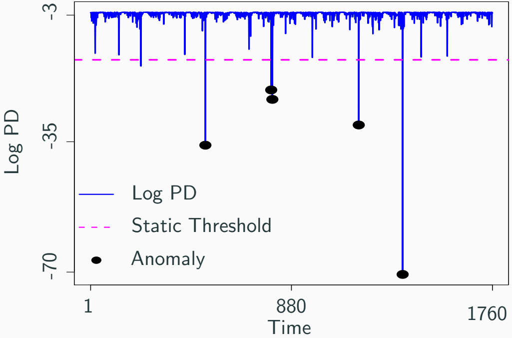
- Experiments conducted with diverse data sets
    - Traffic speed
    - Vehicular occupancy
    - Travel time
    - Taxi demand
    - Biomedical - ECG
    - Finance - Bitcoin
- Proposed EVT-based framework consistently outperforms Gaussian and Tukey's method based strategies

Performance of Detection Strategies



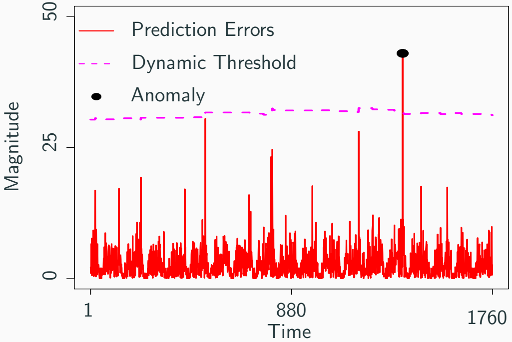
Modeling using LSTM

Performance of Detection Strategies



Gaussian-based detection technique

Performance of Detection Strategies



EVT-based detection technique

End-to-End Deep Anomaly Detection

Algorithm: The training process of the proposed end-to-end EVT-LSTM model. The threshold τ_e is updated every k epochs.

Input: Set of examples $(\mathbf{x}_{n:}, y_{n:}), n : 1, \ldots, N$

Output: Set of decision scores $\varsigma_{n:},\ n : 1, \ldots, N$

Initialization: Threshold $\tau_e \leftarrow 0$

while convergence criteria unmet **do**

  Update weights of the network

  for once in every k epochs **do**

    Calculate prediction errors, $E(\hat{y}_{n:}) = |\hat{y}_{n:} - y_{n:}|$

    $T_{ini} \leftarrow \text{InitThreshold}(E(\hat{y}_{n:}))$

    Excesses $\leftarrow \{E(\hat{y}_{n:}) - T_{ini} | E(\hat{y}_{n:}) > T_{ini}\}$

    Fit a GPD to excesses by using MLE and find parameters

    Update τ_e

Compute decision score $\varsigma_n = |\hat{y}_n - y_n| - \tau_e$ for each $\mathbf{x}_n$

if $\varsigma_n \geq 0$ **then**

  $\mathbf{x}_n$ is anomalous

else

  $\mathbf{x}_n$ is non-anomalous

Proposed End-to-End EVT-LSTM Model

Objective function of LSTM modified as

$$\min_{\mathcal{W}} \frac{1}{N}\sum_{i=1}^{N} || \boxed{\mathrm{E}(\Phi(\mathbf{x}_i;\mathcal{W})) - \tau_e} ||^2 + \frac{\Psi}{2}\sum_{l=1}^{L} ||\mathbf{W}^l||_F{}^2$$

where,

Input: $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ in $\mathcal{X} \subseteq \mathbb{R}^p$, Output: $\{y_1, \ldots, y_N\}$ in $\mathcal{Y} \subseteq \mathbb{R}$

Weights: $\mathcal{W} = \{\mathbf{W}^1, \ldots, \mathbf{W}^L\}$ with L layers & regularization $\Psi > 0$

Absolute Prediction Error: $\mathrm{E}(\Phi(\mathbf{x}_i;\mathcal{W}))$

EVT Threshold: τ_e

- During training, τ_e updated once in every k epochs
- After training, calculate decision score for each $(\mathbf{x}_n, y_n)$ pair

  $\varsigma_n = |\hat{y}_n - y_n| - \tau_e$

  $\varsigma_n \geq 0 \implies x_n$ is anomalous
- **Outperforms** traditional and hybrid deep models on seven diverse data sets

Summary

Contributions

Accurate demand forecasting

can limit surge-pricing by balancing taxi demand and supply

Spatio-Temporal Data → Spatial Aggregation → Temporal Aggregation → Spatio-Temporal Modeling → Demand Forecasts

- Explored fixed and variable spatial partitioning
- No universal winner
- Proposed modified dHEDGE

- Explored various time-series models
- Exploited spatial information
- LSTM variants better than regression models
- Potential of GraphLSTM
- Proposed EVT-based detection rules for anomaly detection
- Superior to the commonly employed detection rules

Publications

Journals, published

1. N. Davis, G. Raina, and K. Jagannathan, "Taxi Demand Forecasting: A HEDGE-Based Tessellation Strategy for Improved Accuracy," IEEE Transactions on Intelligent Transportation Systems, vol. 19, pp. 3686–3697, 2018.

Conferences

1. N. Davis, G. Raina, and K. Jagannathan, "LSTM-Based Anomaly Detection: Detection Rules from Extreme Value Theory," in Proc. of Springer EPIA European Conference on Artificial Intelligence, pp. 572–583, 2019.
2. N. Davis, G. Raina, and K. Jagannathan, "Taxi Demand-Supply Forecasting: Impact of Spatial Partitioning on the Performance of Neural Networks," in NeurIPS Workshop on Machine Learning in Intelligent Transport Systems, 2018.
3. N. Davis, G. Raina and K. Jagannathan, "A Multi-Level Clustering Approach for Forecasting Taxi Travel Demand," in Proc. of IEEE International Conference on Intelligent Transportation Systems, pp. 223–228, 2016.

Journals, under revision

1. N. Davis, G. Raina, and K. Jagannathan, "Grids versus Graphs: Partitioning Space for Improved Taxi Demand-Supply Forecasts," IEEE Transactions on Intelligent Transportation Systems, February 2019.
2. N. Davis, G. Raina, and K. Jagannathan, "A Framework for End-to-End Deep Learning-Based Anomaly Detection in Transportation Networks," Elsevier Transportation Research Interdisciplinary Perspectives, October 2019.