BigQuery-Geotab Intersection Congestion

Final Report

Siyu Li
Master of Computer Science
University of Illinois
Urbana-Champaign
siyul2

Yifei Wang
Master of Computer Science
University of Illinois
Urbana-Champaign
yifeiw16

Yuhan Wu
Master of Computer Science
University of Illinois
Urbana-Champaign
yuhanwu4

Introduction

Intersection congestion is an important aspect of traffic control. Efficient and accurate analysis can help optimize signal control and make the city smarter. From data mining perspective, this project focuses on spatial-temporal forecasting and provides real-life geographical data to explore the relationships between intersection congestion and multiple factors such as day of week and city layout, which is a challenging regression problem. Moreover, its demands to predict multiple quantiles rather than simple averages give us insights into exploring robust feature engineering techniques.

Related work has shown potential in this area. For instance, Recurrent Neural Networks like LSTMs have been proven effective for time-series traffic prediction [4], while Graph Neural Networks (GNNs) has become a powerful tool of modeling spatial data in traffic systems[2].

This project will utilize dataset collected from commercial vehicle telematic devices, with information including locations, specific hours of days, an aggregate measure of stopping distance and waiting times at intersections in 4 major US cities, to anticipate traffic conditions.

The inputs will be the following: the intersection ids, cities, directions that the vehicles are heading for entry and exit, hours of the day, whether the days are weekend or not, and months of the year. With all the information provided above as input, we will need to output three different quantiles (p20, p50, p80) of the total time stopped and the distance between the intersection and the first place a vehicle stopped while waiting.

Two baseline models - linear regression and GCN - will be chosen to predict the traffic condition with the preprocessed data for comparison with our own model: LSTM and XGBoost. The overall performance of all the models will be evaluated with RMSE.

Methodology

Before we start training the data, we need to first implement the data cleaning process to eliminate redundant attributes and standardize attribute parameters.

We first need to check if there is any null value in the dataset to avoid dirty data undermining the training process. After scanning through the dataset, we have found out that there are 8148 null values for *EntryStreetName* and 6287 null values for *ExitStreetName*. As a result, we won't take these two parameters into account during our training process to avoid using null values.

After observing the parameters of the dataset closely, we have divided the attributes into two different types of indicators: location and time.

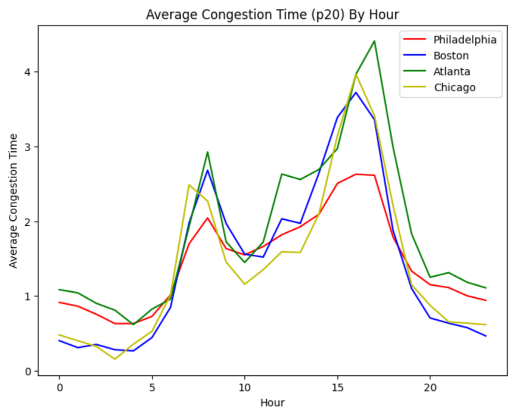
This project will utilize *intersectId* and *City* attributes since using these two attributes together is sufficient to form unique indicators for intersections, so *Rowid*, *Latitude* and *Longitude* are considered redundant attributes. *EntryStreetName* and *ExistStreetName* attributes are also essential because they provide information about the direction vehicles are heading in, and this is an important factor influencing traffic conditions in the real world. Given the fact that *City*, *EntryStreetName* and *ExistStreetName* are categorical attributes, we need to convert them to be utilized in our selected models. The *City* attribute includes four types: Atlanta, Boston, Chicago and Philadelphia, and they are transformed into four binary attributes with variables 0 and 1. Both *EntryStreetName* and *ExitStreetName* have 8 different types that indicate eight different directions, so they are encoded into numerical numbers of 0 to 7.

The time-related attributes include *Hour*, *Month*, and *Weekend*. The *Hour* attribute, which ranges from 0 to 23, represents a cyclical quantity: after hour 23 comes hour 0. Treating *Hour* as a linear numerical variable would incorrectly imply that hour 0 is far from hour 23, which could mislead machine learning models when capturing temporal patterns. To address this, we transform *Hour* into two features using sine and cosine functions:

$$\text{hour\_sin} = \sin\left(\frac{2\pi \cdot \text{Hour}}{24}\right), \quad \text{hour\_cos} = \cos\left(\frac{2\pi \cdot \text{Hour}}{24}\right)$$

This cyclical encoding preserves the circular nature of hours, ensuring that the model recognizes the adjacency of hour 23 and hour 0. It might allows machine learning algorithms to capture daily periodic patterns more effectively, which can be important for predicting time-dependent phenomena in the dataset.

Figure 1: Average Congestion Time vs. Hour



To further explore the relationships between these two different parameter sets, we extracted the attributes *Hour* in time parameter sets and *City* in location parameter sets, and calculated the average congestion time of p20 in relation to these two attributes, as shown in **Figure 1**. The graph has clearly indicated that the average congestion time for p20 varied according to different hours of the day, as we can see that there are two obvious local maximum from 8 to 9 o'clock and from 17 to 18 o'clock. These time slots are considered rush hours, where most people commute for work or school. In addition, we have also created a correlation matrix for the parameters that we selected.

Table 1: Selected Attributes Correlation Matrix

	IntersectionId	EntryHeading	Atlanta	Boston	Chicago	Philadelphia
IntersectionId	1.000	-0.016	-0.425	-0.179	0.383	0.199
EntryHeading	-0.016	1.000	-0.005	0.074	-0.051	-0.020
Atlanta	-0.425	-0.005	1.000	-0.243	-0.201	-0.433
Boston	-0.179	0.074	-0.243	1.000	-0.218	-0.470
Chicago	0.383	-0.051	-0.201	-0.218	1.000	-0.389
Philadelphia	0.199	-0.020	-0.433	-0.470	-0.389	1.000
hour_sin	0.021	-0.010	-0.040	0.078	0.074	-0.086
hour_cos	-0.065	-0.011	0.078	-0.100	-0.181	0.152
month_sin	-0.006	0.002	-0.005	0.030	0.001	-0.022
month_cos	-0.015	0.003	-0.014	0.039	-0.051	0.016
Weekend	-0.077	-0.003	0.086	-0.081	-0.219	0.157

Table 1: Selected Attributes Correlation Matrix (cont.)

	hour_sin	hour_cos	month_sin	month_cos	Weekend
IntersectionId	0.021	-0.065	-0.006	-0.015	-0.077
EntryHeading	-0.010	-0.011	0.002	0.003	-0.003
Atlanta	-0.040	0.078	-0.005	-0.014	0.086
Boston	0.078	-0.100	0.030	0.039	-0.081
Chicago	0.074	-0.181	0.001	-0.051	-0.219
Philadelphia	-0.086	0.152	-0.022	0.016	0.157
hour_sin	1.000	-0.108	-0.012	-0.061	-0.082
hour_cos	-0.108	1.000	-0.035	0.060	0.161
month_sin	-0.012	-0.035	1.000	0.040	0.002
month_cos	-0.061	0.060	0.040	1.000	0.046
Weekend	-0.082	0.161	0.002	0.046	1.000

According to the correlation matrix, the values for all the attributes that we have selected is far from being 1 or negative 1 - which means that none of them is strongly positively correlated or negatively correlated. This indicates that these attributes are not redundant and thus we can train our model using these variables.

After data preprocessing, we start our model training process. We have selected two baseline models, one is linear regression model and the other is GCN. LSTM and XGBoost are chosen to be our approach to outperform the baseline. The detailed modeling process is explained in the Preliminary Results section.

LSTM To model potential temporal dependencies in traffic congestion, we implemented a dual-head Long Short-Term Memory (LSTM) architecture, motivated by prior work showing that LSTM-based models can effectively capture temporal patterns in traffic data [3, 4]. Unlike the baseline models that treat each record independently, the LSTM takes sequences of historical observations and predicts congestion-related quantiles at the next time step. In addition, we later developed a two-stage, zero-aware variant of the LSTM model, where a LightGBM classifier first predicts whether congestion is likely to be non-zero and the LSTM regressor is only applied for such cases. The details and results are reported in the Ablation Study section.

XGBoost We finally tried XGBoost framework [1]. Since the prediction task involves six continuous target variables, we consider the multi-output strategy. To be specific, a single model jointly predicts all target dimensions by wrapping an XGBoost regressor within a multi-output regression interface. Before training, each target dimension is standardized to zero mean and unit variance, ensuring that all outputs contribute proportionally to the optimization objective. Then the predictions are rescaled back to the original space using the inverse transformation.

Experimental Setup

The training set is divided into training samples and validation samples with a ratio of 4:1. Validation samples are randomly selected and follow the distribution of the training set. Models will be trained using training samples, and performance will be evaluated using validation samples.

Linear Regression

The initial baseline model is linear regression model. It is a straight-forward and efficient method to provide a baseline for our own approach to compare with. We used the encoded attributes as mentioned above in the Methodology section to predict the six traffic results.

(1) **Evaluation Metrics:** We will be using the following metrics for evaluating the performance of the linear regression model:
- R^2 **Score:**
$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$
where SS_{res} is the sum of squares of residuals and SS_{tot} is the total sum of squares.

GCN

(1) **Graph Construction:** To incorporate spatial relationships between intersections, we construct a sparse graph using geographic coordinates (latitude and longitude). We employ the *radius_graph* operator from PyTorch Geometric with a 200-meter threshold, approximated in degrees as:
$$r = \frac{200 \text{ m}}{1000 \times 111}.$$
Edges are added between any two nodes within this radius, forming a locality-aware spatial graph:
$$G = (V, E), \quad |V| = N, \quad |E| = \text{edges}.$$
We use *NeighborLoader* to perform mini-batch training with neighbor sampling to reduce memory usage.

(2) **Model Structure:** We adopt a two-layer Graph Convolutional Network for multi-output regression. The network consists of:
- GCNConv layer: $d_{\text{in}} \rightarrow 64$
- ReLU activation
- GCNConv layer: $64 \rightarrow 64$
- ReLU activation
- Fully-connected layer: $64 \rightarrow 6$

(3) **Parameter Settings:**
- Model: 2-layer GCN
- Input features: encoded categorical variables, temporal sinusoidal features, and geographic coordinates
- Hidden dimension: 64
- Output dimension: 6 (multi-output regression)
- Graph construction: *radius_graph* with 200m threshold
- Neighbor sampling: 10 neighbors per layer via *NeighborLoader*
- Optimizer: Adam
- Learning rate: 0.01
- Loss function: Mean Squared Error
- Batch size: 64
- Epochs: 10
- Random seed: 42

Initial implementation of LSTM

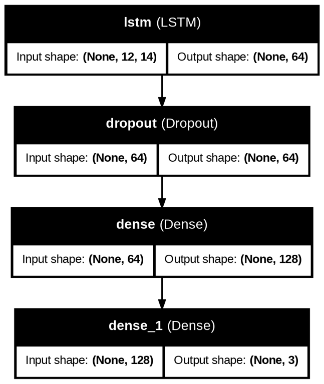
(1) **Sequence construction:** We grouped the data by (City, IntersectionId) so that each sequence corresponds to a single physical intersection. Temporal ordering was reconstructed using cyclical encodings of Hour and Month (sine/cosine). We extracted fixed-length sequences of 12 consecutive observations, where each sequence is used to predict the next traffic state at that location.

(2) **Feature representation:** Each time step contains directional indicators, one-hot encoded city features, cyclical temporal features, and a weekend flag. These features form the sequence input of shape (12,F)(12, F)(12,F), where F is the number of engineered features.

(3) **Model architecture:** Because the dataset includes two distinct groups of target variables—three quantiles for TotalTimeStopped and three for DistanceToFirstStop—we trained two parallel LSTM heads with an identical architecture:
- LSTM layer with 64 units
- Dropout layer (rate = 0.1)
- Dense layer with 128 ReLU units
- Final Dense layer outputting 3 quantile predictions

The TIME head predicts TotalTimeStopped p20, p50, p80. And the DISTANCE head predicts DistanceToFirstStop p20, p50, p80. The overall design of the initial model is illustrated in Figure 2.

Figure 2: Initial LSTM model design



XGBoost

(1) **Target Preprocessing:** The target variables are first standardized using a *StandardScaler* to have zero mean and unit variance. Additionally, a logarithmic transformation (*log1p*) is applied to reduce skewness and improve numerical stability.

(2) **Model Structure and Parameter Settings:** We use *XGBRegressor* as the base estimator, wrapped in *MultiOutputRegressor* for multi-output regression. The hyperparameters of XGBoost are set as follows:
- Objective: *reg:squarederror*
- Learning rate: 0.05
- Maximum tree depth: 10
- Subsample ratio: 0.8
- Column subsample ratio: 0.8
- Tree method: *hist*
- Device: GPU (*cuda*)
- Random seed: 42

(3) **Prediction and Inverse Transformation:** The model predicts on the validation set, producing standardized or log-transformed outputs. Predictions are then converted back to the original scale using the corresponding inverse transformations.

Evaluation Metrics

Following metrics are used to evaluate the model performance:

- **Overall RMSE:**

$$RMSE = \sqrt{\frac{1}{N \cdot d} \sum_{i=1}^{N} \sum_{j=1}^{d} (y_{ij} - \hat{y}_{ij})^2}$$

- **Per-target RMSE:**

$$RMSE_j = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_{ij} - \hat{y}_{ij})^2}, \quad j = 1, \dots, d$$

where N is the number of samples, d is the number of target variables, y_{ij} is the true value, and $\hat{y}_{ij}$ is the predicted value.

Preliminary Results

Baseline-Linear Regression: The performance was evaluated using R^2 score. We also compare the RMSE as shown in Table 2.

Table 2: Validation RMSE of Linear Regression baseline model.

Target	RMSE
TotalTimeStopped_p20	7.11
TotalTimeStopped_p50	15.44
TotalTimeStopped_p80	27.46
DistanceToFirstStop_p20	31.20
DistanceToFirstStop_p50	76.03
DistanceToFirstStop_p80	159.73
Overall RMSE	74.51
R^2 Score	0.023

The linear regression model provides a good baseline for the traffic congestion prediction because it is easy to implement and fast. However, given the fact that we have multi-dimensional attributes and complicated numerical labels, the linear regression model is not competent to give us a strong performance. The R^2 score that we get is 0.022, which is far from 1. Therefore, the result has shown us that the traffic congestion variables might not follow a linear relationship and thus result in poor prediction. In addition, outliers in the dataset might also be an important factor causing the linear regression model to act poorly. A more complicated model is needed for better convergence.

Baseline-GCN: Each sample can be viewed as a point in the map to capture the influence of the traffic from its neighbors, so we construct a sparse graph where each sample is treated as node. The latitude and longitude of each sample are used as node positions and then we set a distance threshold between each node to decide whether the two intersections should build a undirected edge between each other.

Specially, we use radius_graph operator to calculate the distance between transactions and generate the spatial neighborhood adjacency structure.

We employ a two-layer Graph Convolutional Network with a fully-connected regression layer. The GCN layer aggregate information from spatial neighbours using normalized Laplacian message passing. After each convolution layer, non-linear activation Relu is applied. Finally, the last layer maps the learned representations into six dimension, indicating the six parameters the task aims to predict.

The final results are as follows:

Table 3: Validation RMSE of GCN model

Target	RMSE
TotalTimeStopped_p20	7.15
TotalTimeStopped_p50	15.61
TotalTimeStopped_p80	28.26
DistanceToFirstStop_p20	29.48
DistanceToFirstStop_p50	74.59
DistanceToFirstStop_p80	158.79
Overall RMSE	73.87

LSTM: Table 4 summarizes the validation performance of our initial LSTM implementation. Despite being designed to capture sequential patterns, the LSTM model performs worse than both baseline models across all six prediction targets. The RMSE values for the LSTM are substantially higher, particularly for the p50 and p80 quantiles, where errors accumulate over longer-range congestion events. The results indicate that the initial LSTM model was unable to meaningfully leverage temporal information present in the constructed sequences. In contrast to our expectations, the model did not outperform simpler regression-based baselines, suggesting that sequential dependencies in the dataset may be weak, highly irregular, or dominated by noise and zero-inflated targets. A summary of the validation RMSE is shown below:

Table 4: Validation RMSE of LSTM model.

Target	RMSE
TotalTimeStopped_p20	7.50
TotalTimeStopped_p50	16.14
TotalTimeStopped_p80	27.90
DistanceToFirstStop_p20	32.83
DistanceToFirstStop_p50	80.24
DistanceToFirstStop_p80	168.27
Overall RMSE	78.45

Compared to the baseline Linear Regression and GCN models, which achieved considerably lower average RMSE, the LSTM's performance indicates that temporal modeling alone is insufficient for this task. These observations motivated additional investigations into improving the LSTM architecture—through modified sequence construction, zero-filtering techniques, and hyperparameter adjustments.

XGBoost: We employed XGBoost for multi-output regression to predict the six target traffic stop variables. The modeling process is as follows:

We first standardized the target variables using *StandardScaler* to ensure zero mean and unit variance. For the model, we used a base XGBoost regressor with several key hyperparameters: the maximum tree depth (*max_depth*) was carefully selected after experimentation, the learning rate was set to a small value to improve convergence, and both subsample and feature sampling ratios were applied to reduce overfitting. Since there are six targets, we wrapped the regressor with *MultiOutputRegressor* to train one model per target. Predictions on the validation set were then inverse-transformed to the original scale, and performance was evaluated using RMSE.

Table 5: Validation RMSE of XGBoost model.

Target	RMSE
TotalTimeStopped_p20	6.20
TotalTimeStopped_p50	12.05
TotalTimeStopped_p80	20.97
DistanceToFirstStop_p20	28.80
DistanceToFirstStop_p50	62.81
DistanceToFirstStop_p80	120.73
Overall RMSE	57.69

While the XGBoost provides a straightforward and efficient approach, and is the best outcome we have achieved, the overall performance is not particularly strong, especially for the distance-related targets. These results suggest that the model may struggle to capture the complex temporal and spatial patterns in the data, highlighting the need for more sophisticated approaches to better model sequential dependencies.

Ablation study

In order to further experiment with the models we have chosen, we have proposed several experiments by slightly changing the structures and parameters of the models.

LSTM Ablation Study

(1) **Adding extra layers of LSTM**: In order to improve the performance of our LSTM model, we have tried adding an extra LSTM layer with the same parameters. The experimental results are shown below:

Table 6: Validation RMSE of LSTM model with an extra Layer

Target	RMSE
TotalTimeStopped_p20	7.45
TotalTimeStopped_p50	15.92
TotalTimeStopped_p80	27.56
DistanceToFirstStop_p20	32.81
DistanceToFirstStop_p50	79.54
DistanceToFirstStop_p80	165.01
Overall RMSE	77.13

Compared with our proposed LSTM model, adding an extra layer does not improve the performance of. Increasing the complexity of a model does not necessarily improve the performance, as the original LSTM might already be performing well. Adding more model complexity might result in overfitting the model and thus causing the validation error to increase compared with the original model.

(2) **Changing optimizer of LSTM**: The optimizer we used for our model is Adam optimizer, and we did an experiment changing the optimizer to SGD (Stochastic Gradient Descent) and the starting learning rate to 1e-2. The experimental results are shown below:

Table 7: Validation RMSE of LSTM model with SGD optimizer

Target	RMSE
TotalTimeStopped_p20	7.58
TotalTimeStopped_p50	16.57
TotalTimeStopped_p80	29.17
DistanceToFirstStop_p20	32.94
DistanceToFirstStop_p50	81.08
DistanceToFirstStop_p80	171.64
Overall RMSE	79.90

As the results shown, changing the optimizer does not improve the overall performance of the LSTM model. For our dataset, SGD does not work well with our set up and thus using Adam optimizer works better.

Adding Zero-aware Gating

To better handle the strong zero-inflation in the target variables, we developed an improved two-stage "zero-aware" LSTM model. The key idea is to first predict whether a sample is likely to exhibit any congestion at all, and then only apply the LSTM regressor when congestion is likely. Otherwise, the prediction is forced to zero.

(1) **Stage 1: LightGBM-based zero detection** We construct a binary label indicating whether a record exhibits non-zero congestion: $y_{\text{prob}} = (\text{TotalTimeStopped\_p50} > 0 \vee \text{DistanceToFirstStop\_p50} > 0)$
For features, we combine: month one-hot encodings, city one-hot encodings (Atlanta, Boston, Chicago, Philadelphia), and standardized numeric features derived from the remaining input columns.
These features are used to train a LightGBM binary classifier with a logistic loss. We use a learning rate of 0.05, 64 leaves, feature and bagging fractions of 0.8, and a minimum leaf size of 50, training up to 4000 boosting rounds with validation monitoring. The model outputs a probability that a sample is non-zero (i.e., congested).

(2) **Stage 2: LSTM regression on sequences** The regression part follows the LSTM design as the initial implementation, but operates on sequences built in the same feature space as the LightGBM classifier. We first align the feature matrices used for LightGBM Xtrain full,Xval full with the original training and validation indices. We then sort records by (City, IntersectionId, Month, Hour) and group them by city–intersection–month combinations to construct temporally ordered sequences. For each group, we generate sliding windows of length 12; each window contains 12 consecutive observations, and the label is the 6-dimensional target vector at the last time step. The model is trained with MSE loss using the Adam optimizer and early stopping based on validation loss.

(3) **Stage 3: predict based on gating rule at inference time** For each validation sequence, we take the feature vector of its last time step and feed it into the trained LightGBM classifier to obtain the non-zero probability p_{nz}. The LSTM produces a continuous prediction for the same sequence. We then apply a hard gate:

$$\hat{y}_{\text{final}} = \begin{cases} \hat{y}_{\text{LSTM}}, & \text{if } p_{\text{nz}} > \tau, \\ \mathbf{0}, & \text{otherwise,} \end{cases}$$

where we set the threshold $\tau = 0.5$. In practice, this means that if the model believes the sample is unlikely to have any congestion, we directly predict zero; otherwise we rely on the LSTM regressor.

The experimental results are shown below:

Table 8: Validation RMSE of adding a zero-aware Gating for LSTM

Target	RMSE
TotalTimeStopped_p20	6.67
TotalTimeStopped_p50	13.87
TotalTimeStopped_p80	24.51
DistanceToFirstStop_p20	29.45
DistanceToFirstStop_p50	67.94
DistanceToFirstStop_p80	137.01
Overall RMSE	64.67

Implications

Linear Regression According to the R^2 Score of the linear regression model, we can see that the data set does not follow a linear relationship and therefore the performance is not as satisfying as we expected. However, the linear regression model is still a good baseline model since it is fast and simple to implement, and it gives us a basic sense of the metrics we have chosen to be compared with.

GCN The reason for the poor performance of GCN might be the constucted graph does not reflect the true structure of the tracffic condition. The spatial closeness does not necessarily imply actual road connectivity, and multiple time-stamped samples from the same intersection are incorrectly treated as independent nodes.Therefore, there exist substantial noise in the adjacency structure. Moreover, GCNs inherently shmooth features across neighbours, which is unsuitable for discrete attributes such as heading direction, month and hour, resulting the inappropriate broadcast of features. Considering all the factors, the mismatch between the constructed graph and the spatiotemporal nature of the task is the primary cause of the model's limited performance.

LSTM The relatively weak performance of the LSTM model could be caused by the nature of the data set, which does not represent true time-series data. Although each sample contains hour-of-day and month indicators, the observations are not temporally continuous and do not follow a sequential timeline. Therefore, it is difficult for the LSTM model to learn a meaningful temporal trend, as the consecutive rows in the data set do not correspond to the consecutive time in reality.

XGBoost The experimental results show that the XGBoost Multi-Ouput Regressor achieves the strongest performance among all the compared models.This suggests that, for the given dataset, the predictive signal is better captured through the feature-based boosting rather than learned spatial or temporal representations.

Future Work

GCN In terms of graph network, more effort can be put in improving the alignment between the constructed graph and the spatiotemporal character. The modeling in terms of the time can be considered by trying models like T-GCN, which might enable the model to capture the evolution of the traffic patterns over time. What is more, advanced graph attention and large-scale pre-trained traffic models can be implemented to achieve better performance in this task to improve model performance.

LSTM Although our LSTM models struggled to outperform the baseline methods, there are several technical directions that could be explored in future work. A key limitation is that the dataset does not form a true continuous time series, so constructing a more temporally aligned sequence representation may allow the model to capture more meaningful temporal dynamics. In addition, replacing standard LSTMs with lightweight attention mechanisms or transformer-based encoders could help the model focus on relevant time steps without relying on strict recurrence. Finally, instead of using a hard gating rule for zero-inflated targets, a jointly trained multi-task model that predicts both zero probability and continuous values might provide a smoother integration between classification and regression objectives.

XGBoost Although XGBoost makes the best performance, the current framework remains limited in its ability to automatically learn hierarchical or sequential patterns, which neural models are designed to capture. Future work may investigate hybrid approaches that combine tree-based methods with neural encoders, enabling the model to benefit from both representation learning and strong boosting-based regression.

References

[1] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[2] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[3] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic Flow Prediction With Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems* (2015).

[4] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. 2015. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies* 54 (2015), 187–197.

Contributions

Coding

Siyu Li (33%) Data Preprocessing, Baseline Linear Regression, LSTM ablation study

Yifei Wang (33%) Data Preprocessing, Baseline GCN, our own method XGBoost

Yuhan Wu (33%) Data Preprocessing, our own method LSTM, zero-gating LSTM improvement

Report Writing

Siyu Li (33%): Introduction, Methodology, correlation analysis, and sections corresponding to her code implementation.

Yifei Wang (33%): Introduction, Methodology, and sections corresponding to her code implementation.

Yuhan Wu 33%): Introduction, Methodology, and sections corresponding to her code implementation.