Exploring Complex Graph Representations of Highway Networks for Traffic Speed Forecasting

Ricky Miura rmiura@ucsd.edu
Gita Anand gianand@ucsd.edu
Sheena Patel s4patel@ucsd.edu
Gal Mishne gmishne@ucsd.edu
Yusu Wang yuw122@ucsd.edu

Abstract

Traffic speed forecasting, a critical endeavor in urban planning and transportation, requires the construction of sophisticated models capable of navigating the multifaceted dynamics of traffic systems. Current techniques, including machine learning algorithms such as *Random Forest*, and various deep learning approaches (*CNN*, *RNN*), have historically grappled with two primary challenges: the accurate identification of influential features amidst countless variables, and the potential of long-term forecasts. Our research introduces an advanced solution to these issues through the implementation of *Spatial-Temporal Graph Attention Networks* (ST-GAT), which are designed to address the complex spatial-temporal dependencies that traditional methods often fail to capture. Our innovative approach involves the construction of six different graphs, each trained using two distinct ST-GAT models-*ST-GAT Single Edge* and *ST-GAT Edge Type*-differentiated by their handling of edge types. This categorization allows for a detailed scrutiny of how different combinations of node and edge features impact the models' ability to forecast traffic speed with high accuracy, especially over longer time intervals. Through this investigation, we can observe the significance of edge weights and node feature selection in improving predictive performance, thus offering new insights into the plethora of factors influencing traffic dynamics. Preliminary results from our exploration indicate an improvement in the accuracy of traffic flow predictions, particularly for future forecasts. This success not only validates the potential of our ST-GAT framework to revolutionize traffic prediction strategies but also highlights the importance of leveraging graph-based modeling to overcome the limitations faced by existing methodologies. By capturing the nuanced interactions within traffic systems, our work sets the stage for more adaptive urban transportation networks, reflecting a significant step forward in the quest to enhance infrastructure management and planning.

Website:
https://sheenapatel262.github.io/dsc180b_B12_1_website.github.io/
Code: https://github.com/RickyMiura/Traffic-Speed-Forecasting

1 Introduction

In a world where automobiles are everywhere, traffic is an issue that will never go away. Whether it be as a result of a car accident, or as simple as a bottleneck caused by a large volume of cars during rush hour, it is always an inconvenience to drivers. While we can't eliminate traffic due to the sheer volume of automobile use, we can find better ways to predict the flow of traffic. While many navigation systems exist that analyze traffic flow, many of them do not utilize deep learning. Our central question approaching this project was "How can we identify traffic jams earlier on and from greater distances so we can plan our route accordingly?"

In order to combat this, we propose a graph model that emphasizes the relationship between multiple highway sensors within a given traffic dataset, incorporating methods of *time series* analysis and *geo-spatial* analysis. *Graph neural networks* (GNN) are a machine learning data structure, containing *nodes*, entities that hold information, and *edges*, connections between nodes that contain associative pieces of information. A graph is very flexible in that it could be viewed as a way of structuring data, or could be used as a single data point, linked to several other graphs to form a network. This is a major breakthrough, because these relationships can also be included in the implementation of classification, prediction, and other algorithms. The versatility of these data structures are often taken advantage of for deep learning purposes, making them a popular model in recent years. GNNs are particularly advantageous for handling long range interactions, so we believe that this will help improve identifying traffic earlier on.

We will begin with a smaller lens and attempt to access recent traffic sensor data in three of the most traffic-dense highways in San Diego to forecast traffic flow. The ultimate goal of our project is to create a flexible and versatile GNN model applicable to any network of sensors along major freeways where rush hour traffic is prevalent. We will combine the power of GNNs with nuanced features like the number of lanes and other temporal features to optimize a basic linearly-connected GNN model used for traffic prediction. By exploring edge relationships between sensor data, and the impact of different features on accurate traffic flow prediction, we hope to gain a better understanding of what particularly makes a traffic forecasting model successful. Our *baseline* graph highlights the most simple formation out of our many graphs, containing only one edge type and only the speed features. This creates a simplistic linear graph, in which each node, or sensor, is connected to surrounding nodes from the same highway and direction. Our final and most complex graph will feature a combination of three unique edge types, each encapsulating different traffic dynamics, and a multitude of node attributes representing a window of speeds from a given time period, among other temporal features. Finally, we will compare the results between our multitude of graphs to assess the importance of various factors when it comes to traffic speed forecasting.

2 Graph Construction

We investigate various representations of highway networks by developing distinct graph structures, each characterized by unique configurations of *edge connections* and *node features*. For each graph, we generate a series of *graph signals*, with each signal representing the respective graph at different time intervals. This approach allows us to analyze the dynamics of traffic flow across various time frames within the same structural framework.

2.1 Nodes

In our graphs, each *node* represents a sensor positioned along the highway network. We construct a *feature vector* for every node, comprising multiple attributes that reflect the state of the sensor within a specific time frame.

2.1.1 Speed Window

We incorporate the temporal dynamics of a highway by creating, for each node, a *speed window*. This window of size M, comprises average speeds over a range of time, effectively capturing the speed variations over time. The speed at time t is included in the window which means the feature vector at time t will also contain *M-1* previous observations from t. In their paper, Zhang, Yu and Liu (2019) refer to this process of creating a sliding window as *speed2vec*. We can represent the speed window as such:

$$h_t = [v_{t-M+1}, v_{t-M+2}, \ldots, v_t]$$

where t is the current time period, M is the number of time periods we consider including the current time period, and v is the average speed at a given time.

An example of this process can be observed in Figure 1, where we have effectively utilized data collected from a quartet of sensors, explicitly N=4, spanning the time range from 8:45 am to 9:10 am. In this scenario, let us assume that for each interval of five minutes within this specified time frame, we have successfully gathered the velocities as recorded by these sensors, which are then aggregated over each corresponding five-minute segment. Opting for a window size of three, denoted as M=3, enables us to meticulously construct feature vectors that are associated with the time points of 8:55 am, 9:00 am, 9:05 am, and 9:10 am. This is achieved by incorporating information not only from the current time point but also from the two preceding time points, effectively accounting for *M-1* time points. This approach facilitates the construction of a dynamic sliding window of time features, which adeptly captures the nuanced variations in traffic speed observed at previous time steps, thereby providing a comprehensive view of traffic flow dynamics over the selected intervals.

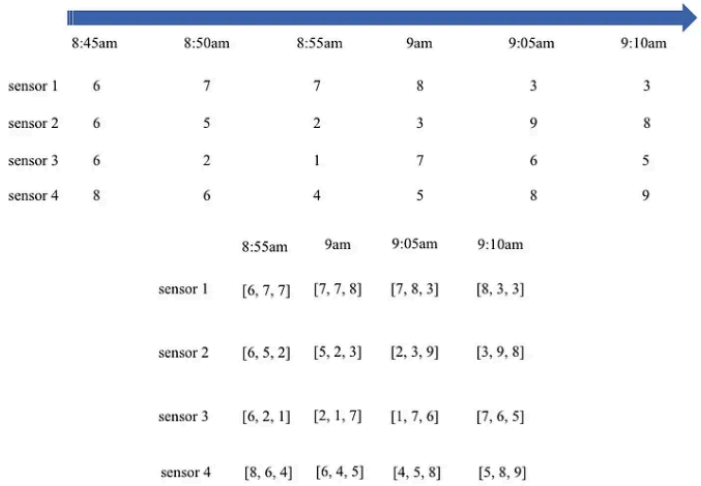


Figure 1: Toy example of the sliding window process (Woodward 2015)

We observe that since we need the previous speeds from two time points, we cannot construct the speed window for the first two time points, 8:45am and 8:50am, without information on even earlier time intervals. We can calculate the number of possible windows we can create in a day as such:

$$\#\text{ of possible windows in a day} = S - (M + H) + 1$$

where S is the number of possible time measurements in a day (288 if measurements were for 5 minute intervals), M is the number of previous speeds to consider including the current time period, and H is the number of future speeds to predict. Note that the windows in Figure 1 do not contain the future speeds in the window and only represent the node feature in the example. When we create the windows, we will include the next H speeds from a given time as well. These future speeds will not be included as a node feature and instead, be used as our target, or *ground truth* observations that we will use to compare our predictions on.

Finally, we can construct a three dimensional feature matrix across all N nodes and across all T time points in the dataset, which has dimensions *[T, M, N]* given by

$$H_N^T = \begin{pmatrix} h_1^1 & h_2^1 & \cdots & h_N^1 \\ h_1^2 & h_2^2 & \cdots & h_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ h_1^T & h_2^T & \cdots & h_N^T \end{pmatrix}$$

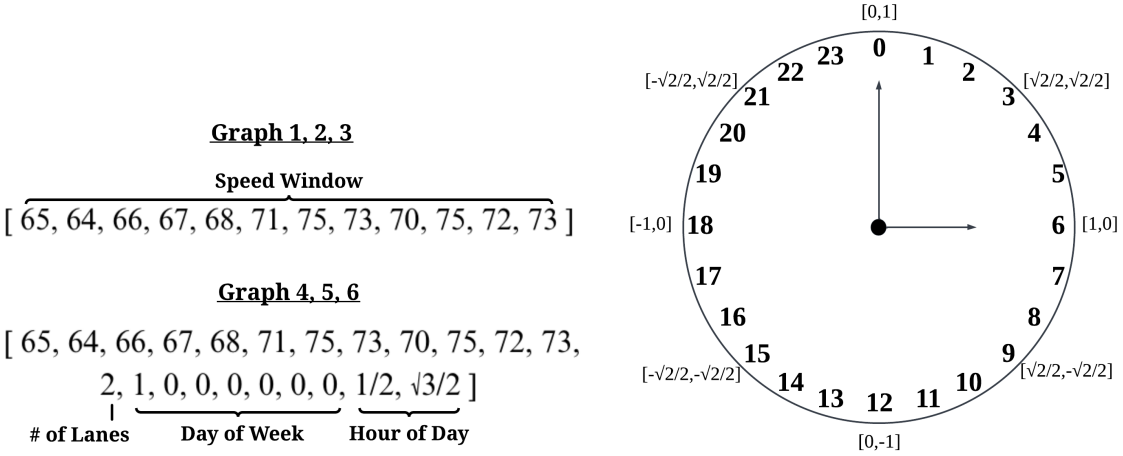
Note that the dimension of length M, which is the size of the window for each entry in the matrix, is not explicitly shown in the above matrix representation.

2.1.2 Lanes, Weekday, Hour

We identify additional features that potentially influence traffic flow dynamics. Among these, the *number of lanes* adjacent to a given sensor is a critical factor. We believe that the lane count directly impacts bottleneck phenomena within traffic streams so we choose to incorporate this into our feature vector and identify any improvement in speed prediction. The lane count is subject to variation across different sensors in the graph; however, for each sensor, this attribute remains constant from one signal to the next as the structure of the graph from signal to signal remains the same.

Additionally, we incorporate the *day of the week* as a critical feature within our analysis, acknowledging that traffic patterns fluctuate significantly across different days, with days such as Friday typically witnessing heightened traffic volumes. To effectively integrate this aspect into our node feature, we employ *one-hot encoding* for the days of the week, thereby augmenting the feature vector's dimensionality by seven—each feature corresponding to a specific day, starting with Monday.

Lastly, we incorporate the *hour of the day* into our feature vector, acknowledging that certain time periods, such as rush hours, witness significantly increased traffic volumes. The hour of the day is initially encoded on a 24-hour scale, where midnight is 0 and 11 p.m. is 23. Recognizing that this linear encoding misrepresents the temporal proximity between late night and early morning hours, we adopt a *sinusoidal* and *cosine* transformation. This approach, shown in Figure 2b, effectively captures the cyclical nature of time, accurately reflecting the closeness between the hours preceding and following midnight. The features representing the day of the week and the hour of the day are consistent across all nodes within a given signal but differ across signals. This variability aligns with our design, as each signal corresponds to a distinct time interval within the unchanged framework of the network structure.



(a) Feature vectors

(b) Cyclical hourly time encoding

Figure 2: Node features

A summary of the node features can be seen in Figure 2a. To evaluate the impact of various feature combinations on model performance, we designate *Graphs 1, 2, and 3* to solely incorporate the speed window as node features. Conversely, we designate *Graphs 4, 5, and 6* to incorporate all the features, including the speed window, lane count, day of the week, and the cyclical representation of the hour of the day. This differentiation allows for a nuanced analysis of how each feature set contributes to traffic speed prediction accuracy.

2.2 Edges

In our graphs, *edges* are indicative of the *spatial proximity* between sensors, thereby integrating the spatial dynamics crucial to understanding the interactions within our highway network. We begin with a *weighted adjacency matrix* that enumerates the *euclidean distances* between every pair of sensors throughout the network. To refine this matrix, we apply a transformation that consists of calculating the *multiplicative inverse* of each element in the matrix. This inversion process effectively assigns greater significance to shorter distances, as data from sensors in closer proximity tend to yield higher accuracy, thereby enhancing the reliability of the information in the context of spatial dynamics. It is important to note that the matrix in question does not contain any zero values, implying a hypothetical link between all sensors irrespective of their designated highways. Utilizing the transformed weighted adjacency matrix, we proceed to apply distinct *masks*, each delineating a specific edge type. These masks are instrumental in fabricating varied connections between sensors, facilitating the handling of diverse road dynamics. We propose three unique edge types and leverage these to generate assorted combinations in the subsequent construction of our graphs, thus tailoring our network models to capture the complexities of traffic behavior more precisely.

2.2.1 Type 1

Type 1 edges construct a linear representation of the network, epitomizing the most immediate spatial connections. For each sensor, an edge is established solely to its closest neighboring sensor in both directions, ensuring a straightforward sequential linkage. This edge type is constrained to the sensor's specific highway, maintaining coherence within each individual network section. Consequently, a sensor positioned on the I8-W would have Type 1 edges connecting it exclusively to the nearest upstream and downstream sensors along the I8-W. Given the intrinsic structure of Type 1 edges, each sensor is limited to a maximum of two such connections, corresponding to its immediate neighbors. Sensors situated at the ends of the highway are an exception, each having a single Type 1 edge that links to the adjacent sensor on their respective ends, thus reflecting the finite boundaries of the network.

Upon applying specified masks to the transformed weighted adjacency matrix, we derive a novel matrix, *W1*, representing inverse distances between nodes where Type 1 edges exist, populated with zeros where such edges are absent. *W1* thus encodes the proximity-based connectivity prescribed by Type 1 edge definition within our network.

2.2.2 Type 2

Type 2 edges are conceptualized to acknowledge potential influences from sensors external to a given sensor's designated highway. Such edges take into account scenarios where congested conditions on one highway could lead to vehicles diverting onto alternate routes. Type 2 edges are therefore intended to encapsulate these minor yet significant traffic patterns, allowing for a more nuanced representation of inter-highway dynamics and their effects on traffic flow. Given the concept of Type 2 edges, these edges are exclusively applied to sensors located outside of a sensor's designated highway. We also establish a distance threshold to ensure connectivity is only among sensors within a specified proximity. This constraint ensures that Type 2 edges represent meaningful relationships between sensors on different highways, capturing the impact of traffic dynamics across nearby routes while maintaining a focus on spatial relevance. To further refine the application of Type 2 edges, we introduce a threshold on the number of such edges that can be established for a given sensor. This measure is particularly crucial in scenarios where numerous sensors are situated in close proximity to each other. The objective of this constraint is to prevent the creation of an excessive number of Type 2 edges, which could potentially dilute the graph with connections that do not convey significant traffic flow information, thereby ensuring that only the most relevant external highway relationships are represented.

Following the application of designated masks to the transformed weighted adjacency matrix, we generate a new matrix, labeled as *W2*. This matrix embodies the inverse distances between nodes interconnected by Type 2 edges, while entries lacking such edges are set to zero. This method ensures *W2* distinctly captures the nuanced connectivity facilitated by Type 2 edges, which are aimed at reflecting the subtle yet critical interactions due to external network influences, all within the scope of our graph's structure.

2.2.3 Type 3

Type 3 edges are conceived to grasp the impact of traffic dynamics originating from more distant locales. This edge type addresses scenarios requiring connectivity to sensors beyond the immediate vicinity within a sensor's designated highway, where direct traversal via Type 1 edges through intermediate sensors would lead to an exponential increase in the number of sensors and edges involved. Such a proliferation poses risks of *over-smoothing* and *over-squashing*, phenomena that can dilute the distinctiveness of traffic patterns and compromise a model's capacity to accurately capture long-range dependencies within the network. To mitigate the challenges posed by the need for extensive propagation through numerous intermediate sensors, we introduce Type 3 edges. These edges function as *skip connections*, enabling the graph to capture *long-range interactions* directly, without traversing every intervening sensor. This edge type is specifically designed to forge connections exclusively within the confines of a given sensor's designated highway, ensuring that Type 3 edges strengthen the depiction of long-range dependencies without crossing into external network territories. Given a specified interval n, the model constructs edges from each sensor to its *2n, 3n, 4n...* sensors along the same highway. For instance, if $n=2$, an edge would be created from a sensor to every second sensor from its position—namely, the second,

fourth, sixth sensor, and so on, emphasizing a systematic approach to capturing extended spatial relationships within the highway network. To mitigate the effects of traffic from sensors located significantly afar, we also implement a distance threshold. This boundary ensures that connections are made selectively, focusing on proximal sensors to maintain the relevance of traffic data influenced by geographical closeness.

After implementing specific masks on the transformed weighted adjacency matrix, a novel matrix emerges, denoted as *W3*. This matrix delineates the inverse distances between nodes linked by Type 3 edges, rendering all other entries as zero. Such a configuration guarantees that *W3* accurately encapsulates the extended connectivity offered by Type 3 edges, which ensure a coherent representation of long-range traffic influences within the specified highway network.

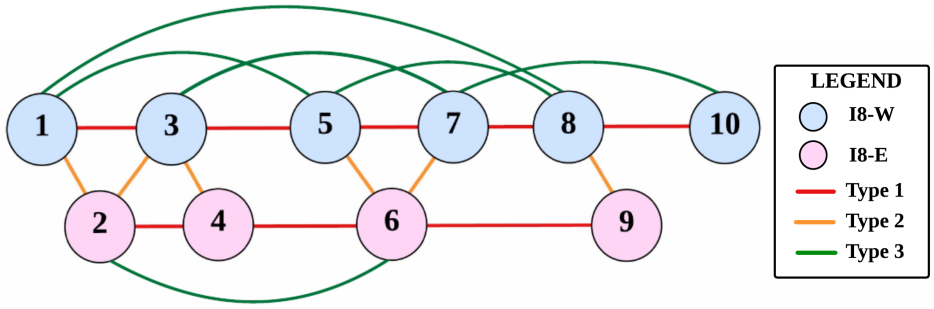


Figure 3: Graph representation of highway depicting different edge types

An example of a graph representation of a network can be seen in Figure 3. As you can see, Type 1 and Type 3 edges are restricted within a given sensor's highway and Type 2 edges are restricted to sensors outside of a given sensor's highway. You can also see certain thresholds coming into play. For instance, due to the distance threshold, no Type 2 edge is established between nodes 6 and 8. Furthermore, with a limit set on the number of Type 2 edges to two per node, node 2 does not form a Type 2 edge with node 5 as it has already reached its quota. Meanwhile, Type 3 edges, operating under an n=2 scheme, facilitate connections such as between node 1 and its second and fourth nearest nodes on the same highway. However, the distance threshold again comes into effect, preventing a Type 3 edge between nodes 3 and 10, despite node 10 being the fourth closest node to node 3, illustrating the nuanced application of these thresholds to maintain a balanced and relevant network model.

Utilizing the specified node features and edge types, we have constructed *six* distinct graphs, each characterized by a unique combination of these elements. A summary of these graphs and their contents can be seen in Table 1.

Table 1: Summary of graphs

Graph	Node Features	Edge Types Included
1	Speed	1
2	Speed	1, 2
3	Speed	1, 2, 3
4	Speed, Lanes, Day of Week, Hour of Day	1
5	Speed, Lanes, Day of Week, Hour of Day	1, 2
6	Speed, Lanes, Day of Week, Hour of Day	1, 2, 3

3 Model Construction

For the purpose of traffic speed prediction, we leverage a *Spatial-Temporal Graph Attention Network* (ST-GAT) proposed by Zhang, Yu and Liu (2019). This advanced model is adept at processing multiple graph signals, each denoting the dynamics of a singular graph structure across different time intervals. Through this approach, ST-GAT efficiently integrates both *spatial relationships* and *temporal variations* within the network, enabling precise and dynamic traffic forecasting. We then propose two variants of the ST-GAT to cater to the nuances of our study. The first variant, *ST-GAT Single Edge*, processes graphs that incorporate various combinations of edge types, yet it does not distinguish between these types, treating all edges as equivalent. Conversely, the second variant, *ST-GAT Edge Type*, operates on identical graph structures but with a key distinction—it recognizes and differentiates among the edge types, assigning unique weights to each. This differentiation allows for a more nuanced understanding of the spatial-temporal dynamics by acknowledging the distinct roles and influences of each edge type within the traffic prediction framework.

3.1 ST-GAT Single Edge

3.1.1 Graph Attention Network

The *ST-GAT Single Edge* initiates its *feed-forward* process with a foundational *Graph Attention* (GAT) layer. GAT builds upon the principles of *Graph Convolutional Networks* (GCN), a foundational *Graph Neural Network* (GNN) architecture. In a GCN framework (Kipf and Welling 2017), the feature vectors of adjacent nodes are aggregated to compute a new feature embedding for a target node. GAT progresses this concept by integrating *self-attention* mechanisms, as detailed in (Veličković et al. 2018), which enables the network to discriminately emphasize certain nodes' features over others during aggregation, thus enhancing the feature embedding with contextual sensitivity and adaptability. This mechanism is crucial for adjusting the influence of neighboring nodes based on their relevance, enhancing the model's predictive capabilities. The schema of a single graph attentional layer is depicted in Figure 4, where a *multi-head* attention mechanism is employed, involving K heads, with N representing the number of nodes connected to node i.

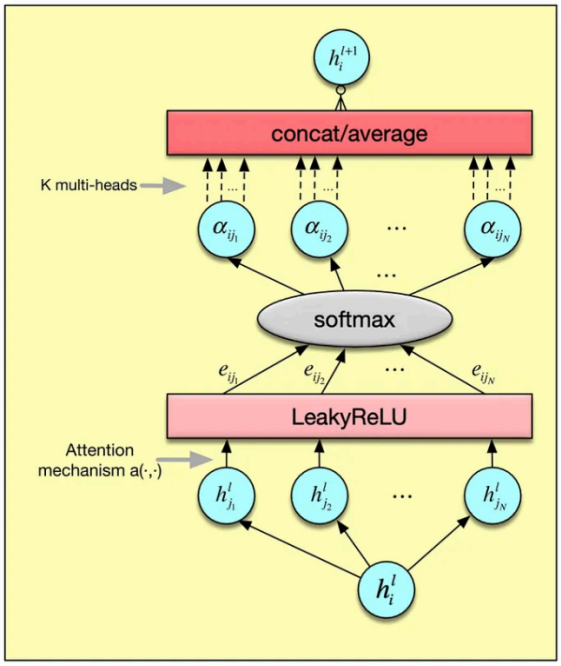


Figure 4: GAT layer (Zhang, Yu and Liu 2019)

For a target node and its neighbors, a Graph Attention Network (GAT) applies a learnable weight matrix and an activation function to compute attention coefficients for each node pair. These coefficients prioritize nodes important to the target's features, offering a flexible aggregation scheme compared to the fixed weights in traditional GCNs.

The input to a single graph attention layer is a set of node features, $\boldsymbol{h} = \{\overrightarrow{h}_1, \overrightarrow{h}_2, ..., \overrightarrow{h}_N\}$, $\overrightarrow{h}_i \in \mathbb{R}^F$ where N is the number of nodes, and F is the number of features in each node. (Veličković et al. 2018) The layer gives us an output which is the set of new node features, $\boldsymbol{h}' = \{\overrightarrow{h}'_1, \overrightarrow{h}'_2, ..., \overrightarrow{h}'_N\}$, $\overrightarrow{h}'_i \in \mathbb{R}^{F'}$, which may or may not be in the same dimension. To obtain stronger expressivity, at least one learnable linear transformation is required so the input features can be converted into a higher-level embedding. The learnable linear transformation is parameterized by a *weight matrix*, $\boldsymbol{W} \in \mathbb{R}^{F' \times F}$, which is then applied to all nodes. The self-attention mechanism that differentiates GAT from GCN is then performed on the nodes-a shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ computes *attention coefficients*

$$e_{ij} = a(\boldsymbol{W}\overrightarrow{h}_i, \boldsymbol{W}\overrightarrow{h}_j)$$

which represents the importance of node j's features to node i. The model assigns every node the ability to focus on every other node, initially ignoring the graph's structure. This is refined by applying masked attention, calculating e_{ij} only for nodes $j \in \mathcal{N}_i$, the neighborhood of node i, primarily consisting of first or second-order neighbors to reduce computational load. Coefficients are then normalized across all choice of j using the *softmax* function to make them easily comparable across different nodes:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp(e_{ik})}$$

After solving for the normalized attention coefficients, the linear combination of the features corresponding to them can be computed to obtain the new embedding of the node features. We could also apply a non-linearity function, σ:

$$\overrightarrow{h'}_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W \overrightarrow{h}_j\right)$$

like *ReLU* or *Sigmoid* activation. Now we have our new and final output features for every node with more expressivity while maintaining some form of the information from the original node feature inputs.

3.1.2 Long Short-Term Memory

The output from the GAT layer serves as the input for a subsequent *Long Short-Term Memory* (LSTM) layer. This sequential processing allows the LSTM to encode the temporal characteristics of the data, thus integrating the time-dependent evolution of traffic patterns into the model's predictive capabilities. The overall framework of a single LSTM layer can be seen in Figure 5.

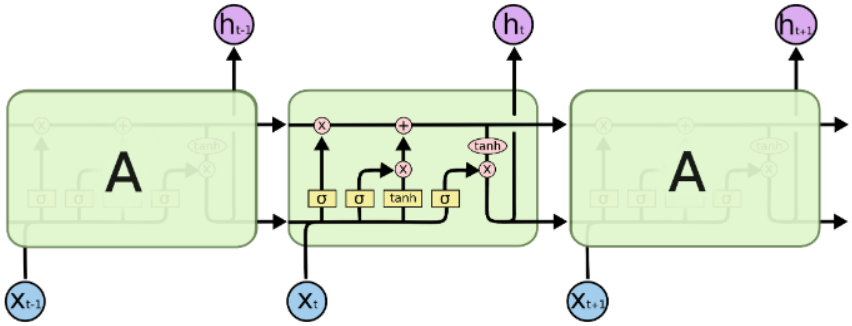


Figure 5: LSTM layer (Olah 2015)

The LSTM architecture is designed to model sequences by overcoming the limitations of traditional *Recurrent Neural Networks* (RNNs) through a complex system of *gates*. Each LSTM unit is composed of a *cell*, an *input gate*, an *output gate*, and a *forget gate*. These components work together to regulate the flow of information both into and out of the cell, as well as the cell's ability to remember or forget information over long periods of time.

At each time step t, the LSTM takes an input vector h_{t-1}, to update its current state h_t and cell state C_t. The LSTM begins by determining what information is to be retained or discarded from the cell state. This decision is facilitated by a Sigmoid layer known as the

"forget gate." The forget gate processes the input x_t and the previous hidden state h_{t-1}, and it outputs a number between 0 and 1 for each component in the cell state C_{t-1}:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Here, f_t represents the forget gate's output, with values close to 1 indicating the retention of the corresponding value in the cell state, and values close to 0 signaling its disposal.

Subsequently, the LSTM decides which new information is to be stored in the cell state. This process is divided into two stages: first, an "input gate layer" (a Sigmoid layer) decides which values will be updated, and then a *tanh* layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

In these expressions, i_t is the output of the input gate, dictating the degree to which each state value is updated, and $\tilde{C}_t$ represents the candidate values proposed for addition to the cell state.

Next, the cell state is updated by combining the old state C_{t-1} with the new candidate values, modulated by the forget gate and the input gate:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

This equation ensures the new cell state C_t is a fusion of past information (as modulated by the forget gate) and new candidate information (as allowed by the input gate).

The final phase of the LSTM's operation involves defining the output. Initially, a Sigmoid layer determines which parts of the cell state will progress to the output. Then, this decision is tempered by the tanh function, which provides a filtered version of the cell state. Finally, the output is computed using the output gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

The function σ denotes the sigmoid activation function which outputs a value between 0 and 1, effectively gating the information. The operator $*$ represents element-wise multiplication. The weights W and biases b are learnable parameters of the model.

By selectively updating and gating information, the LSTM can maintain a memory over many time steps, enabling it to capture long-term dependencies and overcome issues like vanishing gradients which makes LSTMs particularly well-suited for a range of sequence modeling tasks, from language modeling to time series prediction.

In the architecture of our ST-GAT Single Edge model, we incorporate a pair of LSTM layers in sequence. This configuration is designed to enhance the model's capability to apprehend longer temporal dependencies, thereby enriching the sequential pattern recognition within the traffic speed data. Finally, the output harnessed from the second LSTM layer is fed into a *linear output* layer. This layer is responsible for generating the forecasted traffic speeds for H future time intervals, commencing from the current time step t. The linear nature of this layer enables a direct mapping from the high-dimensional LSTM outputs to the predicted speed values, thereby concluding the model's predictive pipeline.

3.2 ST-GAT Edge Type

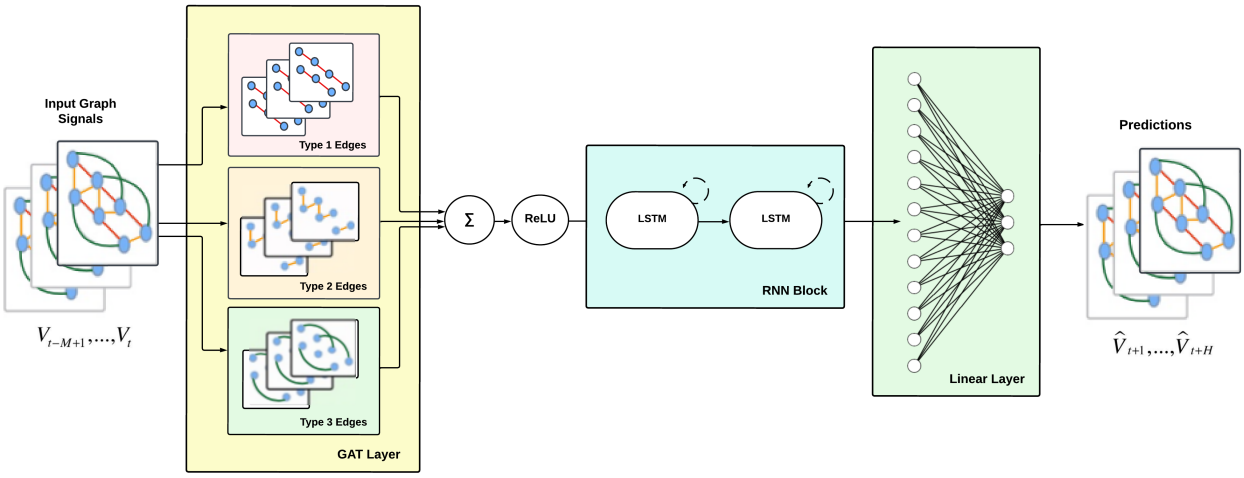


Figure 6: Architecture of ST-GAT Edge Type

ST-GAT Edge Type represents an evolution of ST-GAT Single Edge, equipped with the capability to process identical input graph signals while discerning and learning the importance of various edge types. This model initially *partitions* the input graph according to edge type, thereby generating a distinct graph corresponding to each edge type. This segregation is shown in the GAT layer in Figure 6.

Subsequent to this segregation, the GAT layer is applied to each graph signal set, with each set reflecting a specific edge type. This process is performed independently, facilitating the creation of node feature embeddings that are unique to each edge type. These specialized embeddings are then aggregated, with the sum of the embeddings for each node yielding a comprehensive new feature vector (Schlichtkrull et al. 2017). This composite embedding encapsulates the *heterogeneous* significance of the edge types, yielding a richer representation of node features within the graph.

The remainder of ST-GAT Edge Type parallels the ST-GAT Single Edge architecture. The *multi-edge-type* embeddings, once compiled, are subsequently fed through two LSTM layers. This dual-layer LSTM setup serves to capture the temporal dependencies that the GAT layer cannot, taking into account not just the immediate traffic conditions but also their progression over time. The output from the LSTM layers is then channeled into a linear

output layer. It is this layer that transforms the intricate temporal and spatial information encapsulated in the LSTM output into concrete predictions, providing the anticipated traffic speeds for H subsequent time intervals starting from time t. The linear output layer thus serves as the final step in ST-GAT Edge Type's predictive sequence, translating the graph signals into actionable traffic speed forecasts.

4 Experiments

4.1 Data Collection

Our research capitalized on the database available through the *California Performance Measurement System*, a state-of-the-art platform designed for the real-time surveillance and analysis of traffic conditions throughout California's topography. Leveraging automatic data collection from over 39,000 sensors embedded across the freeway system, PeMS offers a granular view of traffic dynamics, providing insights traffic sensor locations, incident reports, lane counts and closures, toll tag readings, census traffic counts, vehicle classifications, weight-in-motion data, and roadway inventory statistics which we would selectively use in our analysis. In our focused inquiry, we extracted data from sensors pertaining to San Diego County's principal highway networks, the *I-5*, *I-8*, and *I-805*. Access to PeMS required user registration, which we completed to filter and obtain the relevant datasets.

Figure 7: Map showing the sensors we collected data from

The extraction of the desired datasets from PeMS was met with substantial procedural hurdles, primarily due to the system's private API which precluded automated scraping techniques. Our team attempted, both individually and collaboratively, to script an automated extraction process for the CSV data from PeMS. However, recognizing the futility of these attempts, we resorted to the manual procurement of the desired CSV files, each encapsulating traffic sensor data at five-minute intervals. To manage the volume and detail of the data effectively, our investigation was confined to a two-week window.

Table 2: Summary of sensors and edge types

Highway	Number of Sensors
I-5	151
I-805	86
I-8	71

Edge Type	Number of Edges
Type 1	302
Type 2	248
Type 3	5043

The next phase involved the transformation of raw data into a format compatible with graph-based analysis. We derived a weight matrix to illustrate the spatial disposition of the sensor array by calculating the pairwise Euclidean distances using the sensors' longitude and latitude coordinates. These distances were then inverted to form a weight matrix where closer sensors are associated with higher weights, reflective of their stronger connectivity.

Directional connectivity was accounted for through the application of a mask over our weight matrix, ensuring that only sensor pairs on the same highway contributed to the graph's edge weights. Then, a second masking step was employed to simplify the graph structure by limiting connections to immediate neighboring sensors, thereby creating a graph that encapsulated the baseline linear network topology.

Our dataset was thus constituted of a set of distinct graph signals, each representing a five-minute snapshot within our 14-day period of study, from January 1 to January 14, 2024. As the road network's architecture remained unchanged, the graph topology was consistently replicated across all intervals, with nodes representing sensors and edges defined by the weights derived from our preprocessing. The node features were composed of aggregated speed data from the sensor readings, incorporating previous and future time intervals to create a predictive window. This comprehensive framework was leveraged to encode each traffic snapshot into a graph signal, serving as the foundational data structure for our resulting analytical models.

4.2 Setup

Our experimentation begins with the construction of six distinct graphs: Graphs 1 through 6. For each graph, we generate a graph signal that corresponds to each time interval in our 14-day dataset, leveraging data collected from 308 sensors—translating to 308 nodes per signal. The temporal scope of our analysis encompasses the 12 time intervals immediately preceding the current time t, effectively providing a historical context equivalent to the past hour.

Each node within these signals is associated with ground truth labels that represent forecast horizons of 3 time intervals (15 minutes), 6 time intervals (30 minutes), and 9 time intervals (45 minutes) from the time interval represented by the given signal. For graphs incorporating Type 2 edges, we set a threshold limit of 3 edges per sensor and impose a distance constraint of 2 miles, hypothesizing that this distance aligns with the typical spacing of highway exits. In the case of graphs featuring Type 3 edges, the graph is designed to skip 3 nodes and the distance threshold is extended to 5 miles, which we believe accurately reflects the spatial dynamics of traffic influence over longer distances.

Following the construction of our graph signals, we proceed to split our dataset into *training, validation, and test* subsets. To ensure that the model is exposed to traffic patterns characteristic of different days of the week, and to leverage the day of the week feature effectively, we distribute the signals across the subsets in a manner that encompasses the full weekly cycle. Consequently, our training set comprises signals drawn from five-minute intervals spanning from January 1, 2024, to January 7, 2024, ensuring that it encapsulates data from each weekday. The validation set is then constructed from the subsequent three days, designed to fine-tune the model parameters and prevent overfitting. Lastly, the test set is designated to include signals from the final four days of the 14-day span, serving as the ultimate measure of the model's predictive prowess under unseen traffic conditions.

Upon establishing the training configuration, we proceed to initialize the ST-GAT Single Edge and ST-GAT Edge Type models for each of the constructed graphs. We choose a *batch size* of 50 for the training phase of our model, a strategy that optimizes the learning process by balancing computational efficiency with the model's ability to generalize from the training data. Lastly, we choose to train our models for 60 *epochs*. The dimensions of the training data as it passes through the models can be seen in Figure 9.

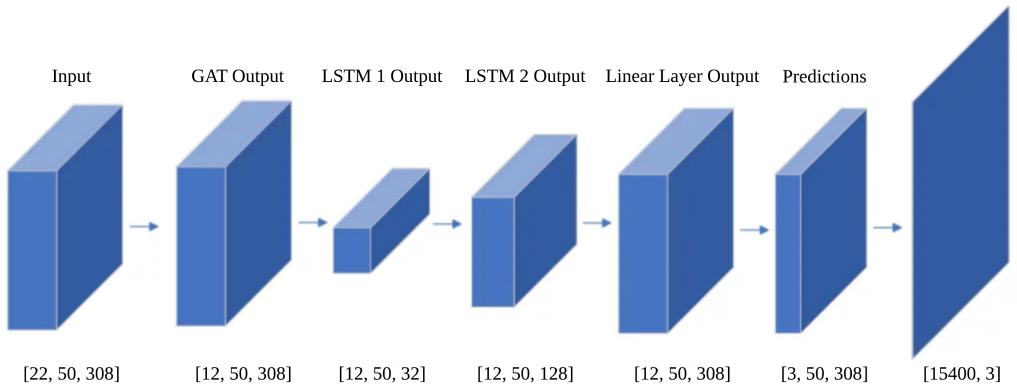


Figure 8: Dimensions through the feed forward process

To put the dimensions in context, we are predicting the traffic speeds for the next 3*5 = 15 minutes based on the previous 12*5 = 60 minutes. The size of each node feature is 22 (size of speed window+number of additional features). The batch size is 50 and we have 308 sensors hence the input dimension is *[22, 50, 308]*. Note that if we only include the speed window as a node feature, the input dimension will instead be *[12,50,308]* but the GAT

output will remain the same. The output of the LSTM 1 and 2 layers are respectively, *[12, 50, 32]* and *[12, 50, 128]* because we choose 32 and 128 as the number of hidden units in each LSTM layer which was found as the best choice of hyperparameters in the experiments conducted by Zhang, Yu and Liu (2019). The prediction tensor is of dimensions *[3, 50, 308]*, since we predict the next $H=3$ time points from the previous $M=12$ time points for each node in all 50 traffic graphs in the batch. We can finally reshape for the purposes of prediction into a two dimensional tensor with dimensions given by *[batch size * num nodes, num prediction time points] = [50 x 308, 3] = [15400, 3]*.

Once our models are trained, we evaluate the performance of the models on the testing set. This stage involves feeding forward graph signals from the testing period through the models to generate traffic speed predictions for future intervals of 15, 30, and 45 minutes. These generated predictions are then compared against the actual ground truth observations. To assess the accuracy and effectiveness of our models, we use three distinct evaluation metrics: *root mean squared error* (RMSE), *mean absolute error* (MAE), and *mean absolute percent error* (MAPE). Their formulas are given below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2} \tag{1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|X_i - \hat{X}_i\right| \tag{2}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{X_i - \hat{X}_i}{X_i}\right| \times 100\% \tag{3}$$

This approach ensures a comprehensive analysis, allowing us to quantitatively measure the models' predictive capabilities in multiple ways.

4.3 Results

Table 3: Evaluations of ST-GAT Single Edge on our graphs

Graph	15 min			30 min			45 min		
	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
1	3.77	2.01	4.67	3.90	2.06	4.83	4.04	2.13	5.03
2	3.89	2.04	4.73	3.97	2.08	4.90	4.07	2.11	5.03
3	3.85	2.01	4.67	3.94	2.05	4.82	4.08	2.15	5.08
4	3.79	2.05	4.68	3.91	2.08	4.87	4.07	2.16	5.06
5	3.91	2.07	4.79	4.00	2.14	5.02	4.16	2.13	5.00
6	3.89	2.03	4.72	4.07	2.09	4.93	4.03	2.06	4.93

Our model's performance in forecasting traffic speeds for imminent 15-minute intervals demonstrated substantial accuracy, surpassing its longer-term predictions. This demon-

strates the model's proficiency in near-term forecasting, showing the potential for immediate traffic management and control applications.

One notable observation was that the incorporation of an extensive feature set did not necessarily translate to enhanced model performance. On the contrary, it resulted in a slight regression, which we suspect is a consequence of time feature over-generalization. Specifically, segmenting time into hourly blocks may have introduced a level of abstraction that overshadows finer temporal nuances crucial for precise prediction, suggesting that the precise temporal resolution is a key determinant in our model's predictive capabilities.

Table 4: Evaluations of ST-GAT Edge Type on our graphs

Graph	**15 min**			**30 min**			**45 min**		
	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
1	3.79	1.98	4.62	3.97	2.03	4.87	3.98	2.05	4.91
2	3.78	2.01	4.69	3.94	2.04	4.89	4.06	2.13	5.06
3	3.87	2.03	4.66	4.11	2.13	5.02	4.11	2.13	5.02
4	3.82	2.02	4.71	3.94	2.03	4.86	4.03	2.12	4.99
5	3.93	2.02	4.75	3.97	2.07	4.86	4.04	2.12	5.01
6	3.85	2.06	4.71	4.01	2.09	4.87	4.10	2.10	5.00

When comparing the performance of different model configurations, our analysis revealed that graphs employing the ST-GAT Edge Type approach yielded superior results compared to those using the ST-GAT Single Edge configuration. This pattern was consistent across all forecasted intervals and indicates the impact of edge differentiation on model efficacy. It also suggests that the ability of ST-GAT models to discern edge types effectively enhances the network, enabling it to capture more complex dependencies and interactions.

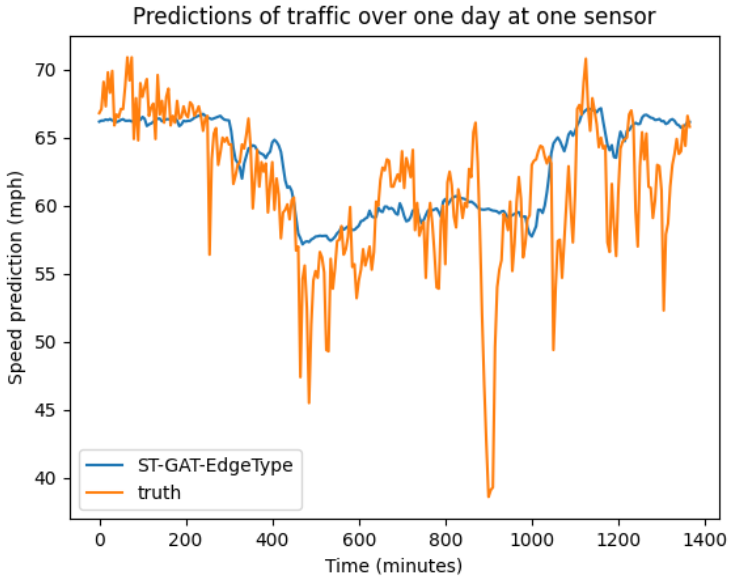


Figure 9: ST-GAT generates smooth prediction

Delving deeper into the edge type performance, graphs with Type 1 edges emerged as the most beneficial to model performance, which is most likely due to the fact that it directly utilized the sensor network's spatial configuration. The proximity of the sensors, as captured by the nearest neighbor speeds, provided a highly dependable foundation for the model to accurately forecast a target sensor's velocity. This finding was backed by our ST-GAT Single Edge model assessments, where models with Type 1 edges consistently outperformed others, as indicated by a lower RMSE, MAE, and MAPE across all timeframes.

Further insights can be found in the evaluation metrics table in Figure 3 and 4. These tables reveal that for the 15-minute predictions, the ST-GAT Single Edge model averaged an RMSE of 3.80, an MAE of 2.03, and a MAPE of 4.71%, while the ST-GAT Edge Type model demonstrated a comparable RMSE of 3.82, but with marginally improved MAE and MAPE values of 2.02 and 4.71%, respectively. This marginal improvement suggests that while the ST-GAT Edge Type models offer an advantage, the ST-GAT Single Edge models are still comparable, emphasizing the relationship between network architecture and predictive capability.

As the forecast interval extended to 30 and 45 minutes, a similar trend was observed, with the ST-GAT Edge Type models maintaining a marginal lead. This could imply that the differentiation of edge types gains more prominence as the forecasting horizon widens, potentially due to the increased complexity of traffic patterns over longer periods, but given the restricted time frame and the magnitude of our data, we cannot conclude this decisively.

5 Ethics and Biases

Our study is rooted in ethical principles, beginning with the responsible extraction and utilization of data from California's Performance Measurement System (PeMS). This system gathers real-time and historical traffic information from sensors spanning the freeway network. While our focus was on a specific group of highway commuters in San Diego, it's important to note that the data collected by these sensors is anonymized, posing no individual privacy risks. Furthermore, in pursuit of enhanced runtime efficiency, we initially concentrated on a subset of the San Diego community. However, our overarching goal is to extend the application of our model to encompass highways across various regions within California. This broader scope ensures equitable treatment of communities and fosters inclusivity in our research endeavors. To uphold transparency and reproducibility standards, we meticulously organized and labeled all datasets and code within structured folders on our GitHub repository. By making our repository publicly accessible, we facilitate reproducibility and enable other researchers to verify or optimize our work.

Acknowledging the inherent biases in the PeMS dataset is imperative. Factors such as sensor placement, time frames, and regional disparities contribute to these biases. For instance, varying distances between sensors result in uneven data density across regions. Although we encountered challenges in rectifying this bias, we adjusted our approach by assigning more weight to closely situated sensors, thereby acknowledging their higher precision. Additionally, our model's reliance on data from only three highways in San Diego limits its

applicability to other freeway networks until further learning is achieved. Our strategy involves retraining the model to accommodate each highway specifically, thereby extending its usability to diverse networks. Moreover, the temporal constraint led to data collection within a limited time frame, potentially skewing year-round predictions. For instance, heightened traffic volumes in January following holiday breaks may distort overall traffic patterns.

In addition to data biases, model bias may arise from subjective hyperparameter choices during training. Parameters such as learning rate, weight decay, and dropout rate significantly impact the model's performance. To address this, we advocate for rigorous parameter tuning using techniques like cross-validation and grid search. Documenting the rationale behind these choices and conducting sensitivity analyses further enhance transparency and reproducibility.

By addressing these biases, we hope that future contributions can take these factors into account by integrating more comprehensive data sets and enhancing model robustness in order to increase the precision of traffic speed forecasts.

6 Future Work

As we conclude our research into traffic flow prediction using graph neural networks (GNNs), there are several aspects in which our work could be further developed and applied given more time and additional resources.

Firstly, we aim to broaden the dataset time frame and streamline the data reconciliation process. Currently, as mentioned before, certain unavoidable restrictions have forced us to manually procure the data, but automation would mitigate human error, given the magnitude of the dataset. We were also restricted in our computing power and manpower on this project, but automation of the data collection process would also allow us to extend the time frame from which we extracted our data beyond the initial two weeks of January 2024 This extended time frame would, in tern, capturing diverse traffic patterns influenced by not only weekends and weekdays, but also national holidays and the the traffic which ensues following the conclusion of a long weekend.

Extending the scope of our input data to integrate additional traffic data, thereby enhancing the performance of our model, is another future step that could be pursued in improving our current model. Our aim entering this project was to analyze traffic flow from three primary highway systems in San Diego City, but in reality, several highways run through San Diego, all of which will affect our traffic forecasting accuracy. Therefore, the inclusion of more highways, could be a way to refine the model, although it will come with the task of obtaining increased computational resources.

Additionally, we are aware that using solely California's Performance Measurement System (PeMS) as our data source may introduce source bias. Therefore, cross-referencing PeMS data with other data sources could increase the robustness of the resulting dataset. Another advantage of using alternate datasets in tandem with our current PeMS data might be to

integrate supplementary data, such as self-report traffic slowdowns, objects reported at certain latitude/longitude coordinates on the highway, or construction at particular times. Acknowledging the cause of traffic at a certain time and having the resulting slowdown at our disposal could potentially improve the model by attributing traffic congestion causes.

While our current research is currently localized to San Diego City, we are aware that different regions hold varying traffic dynamics and feature characteristics, so training our model on highway data within California would allow us to generalize its applicability across multiple contexts. Moreover, by expanding our dataset to be inclusive of local roads surrounding exits, we could capture the traffic on main local roads that could conceivably be the source of slowdowns on a highway in the vicinity through our Type 2 edge construction for a more holistic analysis.

We have investigated different optimization techniques and edge types in our current research, but given an extended time frame for our project, we could also explore ways to advance our architecture by adding layers and optimizing hyperparameters by implementing Grid Search or Cross Validation. Investigating additional feature vector attributes like weather conditions, road work, and event schedules could provide deeper insights, as these elements are known to significantly influence traffic dynamics.

Lastly, embedding real-time feedback mechanisms into our prediction models could facilitate continuous learning and adaptation to changing traffic conditions. Future research could explore the development of reinforcement learning frameworks or online learning algorithms that leverage real-time feedback from traffic sensors to refine model predictions and improve performance over time.

By pursuing these future steps, we aim to advance the field of traffic flow prediction and contribute to the development of more efficient and sustainable transportation systems, benefiting drivers, communities, and society as a whole.

7 Conclusion

In conclusion, our research represents a significant leap forward in the field of traffic speed forecasting, effectively addressing the longstanding challenges that have impeded the accuracy of conventional machine learning and deep learning techniques. By leveraging innovative variants of the Spatial-Temporal Graph Attention Networks (ST-GAT), our study not only transcends the limitations of traditional models in capturing the intricate spatial-temporal dependencies inherent in traffic systems but also illuminates the critical role of edge weights and node feature selection in enhancing predictive accuracy. The implementation of six distinct graphs, each analyzed through two specialized ST-GAT models—ST-GAT Single Edge and ST-GAT Edge Type—has allowed us to evaluate the impact of various node and edge combinations on the models' forecasting capabilities. The findings from our research emphasizes the marked improvement in traffic flow prediction accuracy, particularly for long-term forecasts, thereby underscoring the transformative potential of the ST-GAT framework in revolutionizing traffic prediction strategies.

Furthermore, our work highlights the value of graph-based modeling in overcoming the obstacles faced by existing methodologies, paving the way for the development of more efficient urban transportation networks. The success of our proposed ST-GAT variants not only validates its effectiveness in enhancing infrastructure management and planning, but also sets a new benchmark for future research in traffic speed forecasting. As we continue to refine our models and explore their applications, we anticipate that our contributions will inspire further innovations in the realm of urban planning and transportation, ultimately leading to more sustainable and resilient urban environments.

References

Kipf, Thomas N., and Max Welling. 2017. "Semi-Supervised Classification with Graph Convolutional Networks." [Link]

Olah, Christopher. 2015. "Understanding LSTM Networks." *colah's blog*. [Link]

Schlichtkrull, Michael, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. "Modeling Relational Data with Graph Convolutional Networks." [Link]

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. "Graph Attention Networks." [Link]

Woodward, Amelia. 2015. "Predicting Los Angeles Traffic with Graph Neural Networks." *Medium*. [Link]

Zhang, Chenhan, James J. Q. Yu, and Yi Liu. 2019. "Spatial-Temporal Graph Attention Networks: A Deep Learning Approach for Traffic Forecasting." *IEEE Access* 7: 166246–166256. [Link]