Scalability of Graph Neural Networks in Traffic Forecasting

Assessing Accuracy and Computational Efficiency in Varying Road Network Sizes and Complexities

Danae Natalie Savvidi

Supervisor: Elena Congeduti

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Danae Natalie Savvidi
Final project course: CSE3000 Research Project
Thesis committee: Dr. Elena Congeduti, Prof. Lilika Markatou

Abstract

This paper explores the scalability of Graph Neural Networks (GNNs) in the context of traffic forecasting, a critical area for improving urban mobility and reducing congestion. Despite GNNs' demonstrated effectiveness in handling complex spatiotemporal dependencies in traffic data, scaling them to large road networks remains challenging due to increased computational requirements. This study aims to evaluate how the accuracy and computational cost of a state-of-the-art traffic forecasting GNN, the Decoupled Dynamic Spatio-Temporal Graph Neural Network, change with varying road network sizes and complexities (i.e., sensor density). Using two real-world datasets, three experiments are conducted: scaling map area, scaling graph complexity, and testing the geographic location effect. Findings show that larger graphs generally improve accuracy and GPU efficiency. Moreover, geographic location affects accuracy, whereas sensor density has minimal impact.

1 Introduction

Since the invention of automobiles, people have been waiting in traffic jams. As cities grew, and the number of vehicles increased, this became an even more pressing issue. In recent years, traffic forecasting has emerged as a key strategy to improve the driving experience. It involves predicting the volume, speed, and behaviour of vehicles on road networks over a specific time period. Accurate predictions can lead to more efficient routing, reduced congestion, and improved public safety. The benefits of traffic forecasting attracted government authorities and logistic firms, leading to the development of Intelligent Transportation Systems (ITS) [1].

ITS allowed for the gathering of large amounts of traffic data, creating a need to analyse them effectively. However, even with enough available data, forecasting traffic is inherently challenging due to the complex spatial and temporal dependencies in road networks and the inherent difficulty of long-term forecasting [2]. To address these problems, multiple and increasingly more sophisticated models have been proposed [3] including, more recently, Graph Neural Networks (GNNs).

GNNs emerged as a promising solution for traffic forecasting [3] with recent studies demonstrating their effectiveness in various traffic forecasting tasks, including short-term traffic speed prediction, traffic flow estimation, and congestion detection [4; 5; 6]. Unlike conventional machine learning models, GNNs are specifically designed to handle graph-structured data, making them particularly well-suited for naturally representing traffic networks [7].

Despite these advantages, their ability to efficiently scale to large-scale road networks remains a challenge [3]. As the size and complexity of the graph increase, so do the computational and memory requirements, which can hinder the model's performance and practicality [8]. Multiple sampling-based [9; 10], historical-embedding-based [11], and parallel-computing-based [12] methods have been proposed to address this growing complexity, but research into GNN computational efficiency, accuracy, and ability to handle varying road network sizes in the field of traffic forecasting is still needed [13]. This will help determine if GNNs can be applied in real-world scenarios with continuously growing road networks, and the computational resources they may need to achieve a certain level of accuracy.

This paper's contribution to the state of knowledge is the ***evaluation of the scalability of a state-of-the-art traffic forecasting Graph Neural Network***. Specifically, it investigates how accuracy and computational cost change with variations in road network size, complexity, and geographic location.

The structure of the paper is as follows: Section 2 provides a background on related work, highlighting the evolution of traffic forecasting methods, the development of GNNs, and their scalability problems. Section 3 describes the methodology, including model selection, dataset descriptions, and the experimental design used to assess scalability. Section 4 presents the experimental setup and results, analysing the performance of the selected GNN model across three scenarios. Section 5 discusses the results and limitations of this research. Section 6 reflects on the ethical aspects and reproducibility of the research. Finally, Section 7 concludes with insights gained from the study and suggestions for future work.

2 Background

This section provides an overview of traffic forecasting methods, from traditional statistics to GNNs. Additionally, it formally defines the traffic forecasting problem.

2.1 Related Work

Before Graph Neural Networks

Since the 1970s, various algorithms have been proposed for traffic forecasting, spanning statistical, machine learning, and deep learning methods. Statistical techniques, primarily used for traffic flow predictions, include historical moving average [14], exponential smoothing [15], and autoregressive integrated moving average (ARIMA) models [16]. Non-parametric approaches such as k-nearest neighbours [17], random forests [18], and neural networks [19] also gained popularity for their ability to handle high-dimensional data and learn complex patterns.

Deep learning methods, particularly Graph Neural Networks [3], Convolutional Neural Networks (CNNs) [20; 21], and Recurrent Neural Networks (RNNs), including Long Short-Term Memory networks [22; 23], emerged as promising alternatives for traffic prediction tasks. CNNs can accurately capture spatial dependencies in grid-based traffic data such as New York City, while RNNs excel at learning temporal dependencies. GNNs, however, currently dominate the field due to their ability to model both spatial and temporal complex dependencies.

Graph Neural Networks and Scalability Problems

Despite the promise shown by GNNs in capturing the spatial and temporal dependencies in road networks, their ability to scale efficiently remains a challenge [3]. Scalability has consistently been an issue when developing models for traffic forecasting. Statistical methods, though simple and fast, struggle with processing multivariate data common in large-scale traffic networks, resulting in lower accuracy. Furthermore, machine learning and deep learning models lead to high computational complexity w.r.t. dataset size [24]. GNNs, in particular, require quadratic computational complexity w.r.t. the number of sensors [25]. Addressing the scalability issues of GNNs involves understanding the reasons behind their computational complexity.

Graph Neural Networks use node and edge embeddings to represent node features and relationships within a graph, respectively. Additionally, an adjacency matrix stores the node connections in the graph. This enables GNNs to predict properties for specific nodes (node-level), connections (edge-level), or the graph as a whole (graph-level). Their key mechanism is message passing and aggregation. During each layer, nodes exchange information with their neighbours to update their node representations. The result is that as the number of nodes and edges increases, the amount of operations needed makes computing the new node embeddings infeasible and impractical. Moreover, storing the feature representations and the adjacency matrix for a large-scale graph demands substantial GPU memory [8]. To overcome these issues, recent research has focused on developing efficient techniques and architectures.

Current Research in Graph Neural Network Scalability for Traffic Forecasting

An intuitive strategy for addressing scalability involves sampling specific nodes, a successful approach designed for static graphs [8]. However, in traffic forecasting, the complexity increases further due to temporal dependencies. Spatiotemporal graph neural networks (STGNNs), designed specifically to model graphs with time-varying signals, have increasingly become a preferred solution to traffic forecasting [25].

Existing scalable STGNN architectures rely mainly on sampling or precomputing techniques [26]. Sampling methods have recently been modified and applied to solve time series forecasting in large-scale graphs [27; 28]. However, they struggle to retain the structural information of the graph, and can fail to capture long-term spatiotemporal dependencies. Precomputation techniques, seen in the Scalable Inception Graph Network [29], decouple the graph convolution operations from the training process. Graph convolutions for node features are precomputed and stored to be later used for training. The Scalable Graph Predictor [30] applies this concept to STGNNs. Precomputation techniques however can be less effective due to their fixed representations, greater reliance on hyperparameter tuning, and potential lack of adaptability to new data patterns [26].

Despite these advancements, there's a noticeable gap in research regarding the scalability of STGNNs [30]. Before proposing new models to solve scalability issues, it is crucial to evaluate the scalability of existing traffic forecasting models and identify areas for improvement. Presently, however, there's a gap in such evaluations. While new studies often compare existing traffic forecasting models, they tend to use the same datasets without introducing variations that could test the models' scalability. Notably, there has been only one large-scale study on traffic forecasting [31], and to the best of the author's knowledge, no study systematically scales the size and complexity of road networks to observe corresponding performance changes for STGNN models. Such evaluations could provide insights into STGNN scalability, potentially leading to the development of more robust and adaptable models in traffic forecasting.

2.2 Formal Problem Description

Graph Neural Networks leverage graph structures to represent road networks, where nodes are sensors placed along the roads and edges denote connections between them. Each node v_i is represented by a feature vector at time t, which includes data of dimension C. In this study, $C = 1$ as only speed is measured. Traffic speed forecasting can be defined as a node-level task. Definitions of a traffic network, traffic signal and the traffic forecasting problem are provided below. The definitions follow [32].

Definition 2.1 (Traffic Network)**.** A traffic network is a directed or undirected graph $G = (V, E)$, where V is the set of N nodes (sensors) and E is the set of edges. The reachability between nodes, expressed as an adjacent matrix $A \in \mathbb{R}^{N \times N}$, is determined by the pairwise road distances between nodes.

Definition 2.2 (Traffic Signal)**.** The traffic signal $\mathbf{X}_t \in \mathbb{R}^N$ denotes the speed observation for all sensors on the traffic network G at time step t.

Definition 2.3 (Traffic Speed Forecasting)**.** Given historical traffic signals $\mathcal{X} = [\mathbf{X}_{t-T_h+1}, \ldots, \mathbf{X}_{t-1}, \mathbf{X}_t] \in \mathbb{R}^{T_h \times N}$ from the T_h past time steps, traffic forecasting aims to predict the future traffic signals $\mathcal{Y} = [\mathbf{X}_{t+1}, \mathbf{X}_{t+2}, \ldots, \mathbf{X}_{t+T_f}]$ of the T_f nearest future time steps.

3 Methodology

An overview of the methodology is illustrated in Figure 2. Initially, the model and datasets are chosen. These datasets are then sampled to create four scenarios, which are utilized in the three experiments detailed in this section. Additionally, the metrics employed are explained.

3.1 Model Selection

Models Considered

Choosing a state-of-the-art model to test the scalability of Graph Neural Networks is crucial for drawing valid conclusions. This study considered three models.

The Diffusion Convolutional Recurrent Neural Network (DCRNN) [2] was initially appealing due to prior research and reproducibility [33], offering a head start in implementation and comparative data. However, its computational demands were considerably higher than for the other models.

The Spatio-Temporal Graph Mixformer stood out for its superior performance [34]. However, it is a mixformer with a transformer-based architecture, designed for sequence modelling tasks, whereas GNNs specialize in learning from graph-structured data. As this study focuses on the scalability of GNNs, using a mixformer as a model would not accurately reflect the characteristics of GNNs.

The Decoupled Dynamic Spatio-Temporal Graph Neural (D2STGNN) [32] appeared to be the most suitable choice,

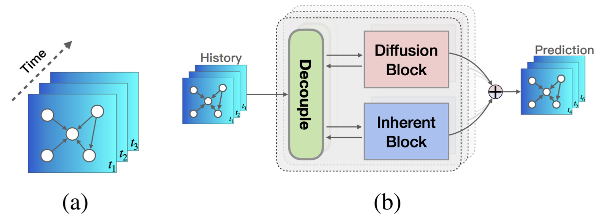


Figure 1: Graph-structured data (Figure 1a) and the Decoupled Spatial-Temporal Framework (Figure 1b). Source [32].

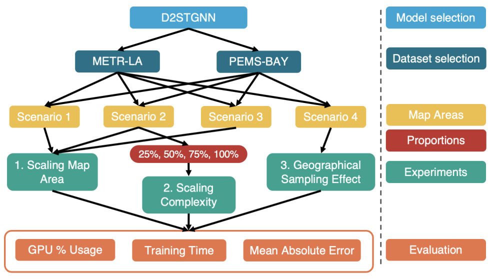


Figure 2: Overview of the methodology used. First, a GNN model, the Decoupled Dynamic Spatio-Temporal Graph Neural Network, and two datasets, METR-LA and PEMS-BAY, are selected for the experiments. Then, for each dataset, four experimental scenarios are created by sampling specific sensors. In Experiment 2, additional subsampling is applied to Scenario 2. Finally, the results of each experiment are evaluated using three main metrics.

as it outperformed in accuracy most other models in a large-scale traffic forecasting benchmarking study [31]. Moreover, its implementation proved reliable, and its training time was significantly shorter than that of DCRNN's.

Decoupled Dynamic Spatio-Temporal Graph Neural Network

The D2STGNN model [32], is designed to better predict traffic patterns by distinguishing between two types of traffic signals: inherent and diffusion. Inherent signals are local, originating near a sensor, while diffusion signals come from other areas in the network. The model takes as input graph-structured data shown in Figure 1a and uses a Decoupled Spatial-Temporal Framework (Figure 1b) to model the two signal types separately. A detailed illustration of the model and signals can be seen in Figures 17, 18 of Appendix C.

The model begins by transforming raw traffic signals into a latent space using a linear layer. It then processes these signals through a decoupling mechanism that separates them into diffusion and inherent signals using a residual decomposition mechanism and an estimation gate. The former decomposes traffic signals by removing parts of the signals that are well-approximated by either the diffusion or inherent models, and retains the rest for further processing. The latter estimates the proportion of diffusion and inherent signals in the traffic data. The diffusion block then uses spatial-temporal localized convolutional layers to model the impact of neighbouring nodes, while the inherent block uses Gated Recurrent Units and multi-head self-attention layers to capture short-term and long-term dependencies, respectively.

Additionally, the model utilizes a dynamic graph learning module to capture how traffic flow relationships change over time. Multiple decoupled spatial-temporal layers are stacked to capture complex patterns, and final predictions are generated by a regression layer. The model is trained using the mean absolute error (MAE), as defined in Equation 1.

3.2 Datasets

The experiment is conducted on two real-world datasets commonly used for traffic prediction. They contain data on the average traffic speed of the vehicles. Due to the speed limit in the areas, the speed value is a float with a range usually between 0 and 70 miles per hour [33].

1. METR-LA (Figure 3a): collected from the highway of Los Angeles County, spanning March 1 to June 10, 2012, and includes data from 207 sensors recording average speed every 5 minutes, totalling 6,519,002 observations.
2. PEMS-BAY (Figure 3b): collected by California Transportation Agencies Performance Measurement System in the Bay Area, spanning January 1 to May 31, 2017. It includes data from 325 sensors recording average speed every 5 minutes, totalling 16,937,179 observations.

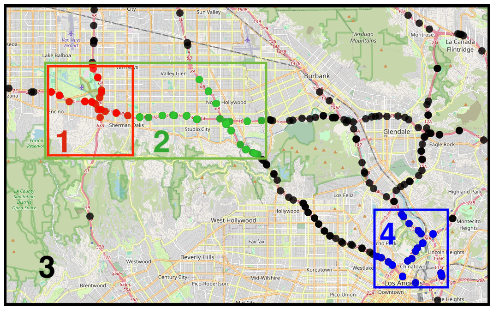
The selection of datasets was based on the following three factors; size (number of sensors, map area covered, time span), type (e.g. speed), and prior use in evaluating D2STGNN in other studies. Initially, datasets used in the original D2STGNN paper [32] were considered. Specifically, PEMS04 and PEMS08, which contain traffic flow data from the San Francisco Bay Area. However, traffic speed prediction, having been more extensively researched [3] was the better choice. The METR-LA and PEMS-BAY datasets [2] were selected for this purpose. These datasets from two different areas in California allowed for better research into how scalability is affected based on road structure and time span of data. Additionally, the CA dataset introduced in a large-scale traffic forecasting benchmarking study [31] was considered but was not selected due to its extensive size and limited current computational resources.

3.3 Scalability Study

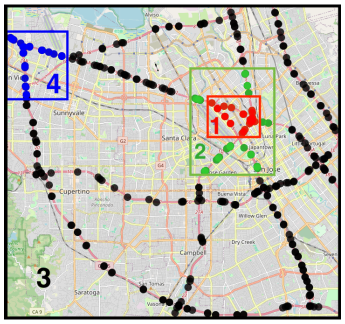
To assess the model's scalability, three experiments were designed, drawing inspiration from a similar study on the DCRNN model [33]. Experiment 1 builds upon the study's approach of comparing accuracy between the full dataset and a subset. Experiment 3 investigates the impact of geographic location on model accuracy, similar to the DCRNN study, but with an additional focus on efficiency metrics.

Experiment 1: Scaling Map Area

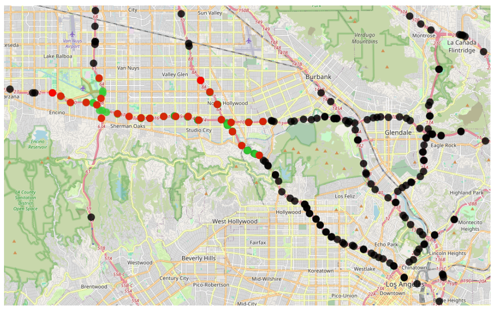
This experiment aims to observe performance changes as the map area grows. To achieve this, a baseline map size is established, and the map area is progressively increased. This results in three scenarios for each dataset visualized in Figures 3a and 3b, with the number of sensors sampled in each scenario shown in Table 1. Specifically, **Scenario 1 (small)** involves a small subset of the original dataset, **Scenario 2 (large)** expands Scenario 1 to include additional sensors, and



(a) METR-LA.



(b) PEMS-BAY.



(c) Example of scattered sensors, 50% proportion.

Figure 3: Sensors selected for the experiments. In Figures 3a, 3b Scenarios 1, 2, 3 and 4 are represented in red, red and green, blue, and all colours respectively. Figure 3c shows an example of scattered sensors selected with 50% proportion (in red) from the Scenario 2 sensors (in green and red) in the METR-LA dataset.

Scenario 3 (original) encompasses the full dataset with all sensors. Scenario 4 (comparison) is used and explained in Experiment 3.

The subsets were chosen based on two factors: size and geographic location. For the PEMS-BAY and METR-LA datasets, sensors within and outside the city were selected respectively. Additionally, a similar number of sensors for the areas across the datasets was ensured, to allow for a fair comparison of trends. One notable difference is how in the METR-LA dataset, the smaller area grows into the larger area along the x-axis, while in the PEMS-BAY dataset, the growth occurs along both the x and y axes. This is to observe whether the direction of growth influences performance.

Table 1: Number of sensors sampled in each scenario. The samples used for Experiment 1 are highlighted.

Scenario	Proportion %	Number of Sensors	
		METR-LA	PEMS-BAY
(1) Small	**100**	**24**	**24**
(2) Large	25	14	13
	50	28	26
	75	42	39
	100	**55**	**51**
(3) Original	**100**	**207**	**325**
(4) Comparison	100	24	24

Experiment 2: Scaling Graph Complexity

The objective of this experiment is to evaluate how the model scales as graph complexity increases, and the map size remains constant. This means increasing sensor density in the map. This is highly relevant for real-world scenarios; understanding how the model performs with varying sensor densities can reveal whether sparser sensor placement could improve training time and accuracy.

In this experiment, sensors are subsampled from Scenario 2 with varying proportions, namely 100%, 75%, 50%, and 25% of the original number of sensors in that area. To better capture the spatial characteristics of the area, points are selected with maximum scattering. This is done using an approximation algorithm[1] outlined in Algorithm 1 as the high sensor count renders the greedy approach computationally infeasible. This algorithm computes the convex hull of the dataset, selects up to M maximally distant points from it, and iteratively adds interior points that maximize their distance until M points are chosen.

The exact number of sensors selected in each proportion is shown in Table 1 under Scenario 2. An example of this sampling is shown in Figure 3c, where a sampling (in red) of 50% is applied on the original sensors in Scenario 2 (in green and red) for METR-LA.

Algorithm 1 Select n Most Scattered Points (Approximation)

Input: List of points $points$, integer n
Output: The selected points

```
1: Compute the convex hull of $points$
2: $selected\_points \leftarrow$ list of points on the convex hull
3: while $len(selected\_points) < n$ do
4:     Find a point $p$ in $points$ that maximizes its distance from the $selected\_points$
5:     $selected\_points \leftarrow selected\_points \cup \{p\}$
6: end while
7: return $selected\_points$
```

Experiment 3: Geographic Location Effect

To investigate whether the geographic location affects the results, **Scenario 4 (comparison)** samples 24 sensors from a different area than Scenario 1 and the two are compared. For METR-LA sensors closer to the city are chosen, and for PEMS-BAY, sensors outside the city but still near another airport. Airports can impact performance in a D2STGNN, as more inherent signals are expected in such areas. With this choice, the aim is to mitigate this impact and focus on the effect of the road structure. The sensors selected for Scenario 4 for each dataset can be seen in blue in Figures 3a and 3b.

3.4 Evaluation Metrics

The evaluation and comparison of the D2STGNN model in each scenario considers two aspects: accuracy and efficiency.

[1]https://scicomp.stackexchange.com/questions/20030/selecting-most-scattered-points-from-a-set-of-points

To assess model accuracy, three common metrics in traffic forecasting were considered: MAE, root mean squared error (RMSE), and mean absolute percentage error (MAPE). MAE was chosen over RMSE for its interpretability and robustness to outliers. MAPE can be problematic for traffic forecasting, as broken sensors yield zero or near-zero values. Moreover, since the sensors are placed on highways with uniform speed limits, using MAE should not impact result comparability.

The MAE measures the average absolute errors across all sensors and future time steps. It quantifies the difference between the predicted traffic signals $\hat{\mathcal{Y}}$ and the actual traffic signals $\mathcal{Y}$, as defined in Equation 1, adapted from [32]. Its domain spans $\mathbb{R}^+$, and lower values signify higher accuracy. In this study, the MAE values presented are measured over the test set.

$$\text{MAE} = \frac{1}{T_f \cdot N} \sum_{t=1}^{T_f} \sum_{i=1}^{N} \left| \hat{\mathcal{Y}}_{ti} - \mathcal{Y}_{ti} \right| \tag{1}$$

The model's efficiency is assessed using the total runtime in seconds (training plus inference time) and average GPU % utilized[2] during runtime. Additionally, training time per node measured in seconds (Equation 2) is used to assess the effect of parallelization techniques.

$$\text{Training Time per Node} = \frac{\text{Total Training Time}}{N} \tag{2}$$

4 Experimental Setup and Results

This section outlines the experimental environment, data preprocessing, and training approach, followed by a detailed description of the results for each experiment.

4.1 Experiment Setup

Environment and Code Implementation

The experiments were conducted on a Windows 11 computer with 64GB of RAM, a NVIDIA GeForce RTX 4090 GPU, and an Intel i9-13900K CPU. The code for the model was obtained from the official D2STGNN repository [35].

Training and Data Splitting

The same data splits and preprocessing of previous work is used [32]. In particular, the datasets are chronologically split into training, validation and testing sets with a ratio of 7:2:1. The adjacency matrix is obtained by applying a thresholded Gaussian kernel on the pairwise road distances among sensors. Sequence samples are generated via a sliding window approach with a width of 24 time steps (equivalent to 2 hours). The first hour is served as input, while the remaining hour is the ground truth. Forecasting is conducted at 5-minute intervals (equivalent to 12 horizons), and the model's performance is assessed by averaging across all horizons for the MAE metric. Figure 4 shows a visualization of this approach. Finally, the model parameters are the same as in the original implementation, except for the warm-up epochs in PEMS-BAY, which is set to 0 to match METR-LA's setting. The batch size is kept at 32, and the epoch count at 80.

[2]https://docs.wandb.ai/guides/app/features/system-metrics#gpu-utilization

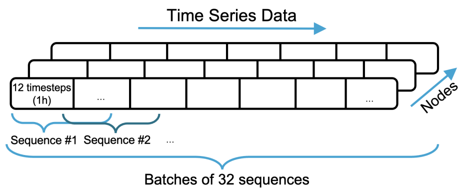


Figure 4: Overview of training approach. Samples are generated using a 24-timestep sliding window, with 12 time steps as input and 12 as ground truth.

4.2 Results

Broken Sensor Values

Broken sensors in the dataset result from malfunctions during data collection. Their values can negatively impact model training and evaluation. To mitigate their impact during evaluation, zero values are ignored when calculating the metrics. However, during the experiment, MAPE values sometimes abnormally reached negative values. It was discovered that broken sensors were not filtered out correctly, with their true value being $-3.8146973 \times 10^{-6}$ in cases due to preprocessing steps in the provided implementation. The metrics in the code were modified to ignore broken sensor values correctly during evaluation.

Experiment 1 Results: Scaling Map Size

Figure 6 illustrates the MAE over the running time, measured after each training epoch, for all scenarios and datasets. In this experiment, Scenarios 1, 2 and 3 are considered. Initially, the MAE decreases rapidly, with the improvement rate slowing down as training progresses. Eventually, larger graphs achieve lower MAE than smaller graphs for both datasets. The lowest MAE values achieved in each scenario are shown in Figure 7a. Large graphs achieve performance levels close to optimal quickly, considering their size. Normalizing training times by node count (Figure 7b) for a more equitable comparison, reveals that larger graphs require less training time per node and maintain higher accuracy. Further analysis shows larger graphs utilize the GPU more efficiently (Figure 7c), and have a higher average node degree (Figure 7d).

The improved accuracy in larger graphs prompts an investigation into whether this applies to every sensor within the network. To explore this, two sensors that are common to small, large and original scenarios were randomly sampled: one on the border of the small and large area (in purple), and another on the south road of the small area (in blue), shown in Figure 5. The results, portrayed in Table 2, indicate that larger graphs for both sensors generally yield slightly higher accuracy, though this is not always the case. The predictions of the small and large graphs over the same sensors (Figures 12, 11 in Appendix A), illustrate that predictions on the larger graph tend to be more stable than the small one.

Experiment 2 Results: Scaling Complexity

In contrast to Experiment 1, where the number of nodes increases with the map area, Experiment 2 maintains a fixed area while varying the number of nodes. This involves proportionally selecting subsets of sensors from Scenario 2 to examine the quantitative relationship between performance

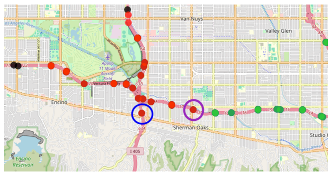


Figure 5: Sampled sensors in METR-LA Scenario 1 for Experiment 1, with IDs 717816 (Blue) and 717499 (Purple).

Table 2: Mean absolute error (MAE) of sensors in Figure 5.

Sensor ID	Scenario	MAE
717816 (Blue)	(1) Small	3.51
	(2) Large	3.33
	(3) Original	3.35
717499 (Purple)	(1) Small	4.72
	(2) Large	4.64
	(3) Original	4.09

metrics and sensor count. The results of the experiment are illustrated in Figure 8. For METR-LA, the MAE decreases for all proportions, with a sharp drop at around 18 minutes, after which it slowly stabilizes. Higher proportions (75%, 100%) result in slightly lower MAE. In PEMS-BAY, the MAE has a similar trend with the MAE decreasing rapidly in the first 16 minutes, and with lower proportions (25%, 50%) showing lower MAE values. The learning speed is similar across all proportions within each dataset. Figure 9b displays the best MAE values achieved for each proportion in the datasets, while Figure 9a illustrates an increase in GPU usage the proportion of the nodes used grows, for both datasets.

Experiment 3 Results: Geographic Location Effect

In this experiment, Scenarios 1 and 4 are compared. Their runtimes, node count and GPU usage (Figures 6, 7c) are nearly identical, enabling a comparison of their accuracy. Scenario 4 shows slightly lower accuracy than Scenario 1 in METR-LA, while the opposite is observed for PEMS-BAY. Experiment 1's findings showed lower MAE in larger graphs, suggesting Scenario 3 should outperform its subgraph, Scenario 4, in both datasets. However, MAE results for PEMS-BAY Scenario 4, do not align with this expectation.

Geographical region is a key factor influencing accuracy, but missing values (broken sensors) also play a role [36]. After calculating the percentage of missing values in the training set of each scenario (Table 3), it's clear that Scenario 4 of PEMS-BAY has a significantly lower ratio of missing values than Scenario 3, resulting in improved accuracy. Nonetheless, this finding does not discredit other results, as the percentage of missing values in other scenarios remains similar.

5 Discussion

This section interprets and verifies the experimental results, and discusses this study's limitations.

5.1 Interpretation of Results

Scaling Map Size

The results from Experiment 1 indicate that larger graphs achieve better accuracy, likely due to nodes in these graphs

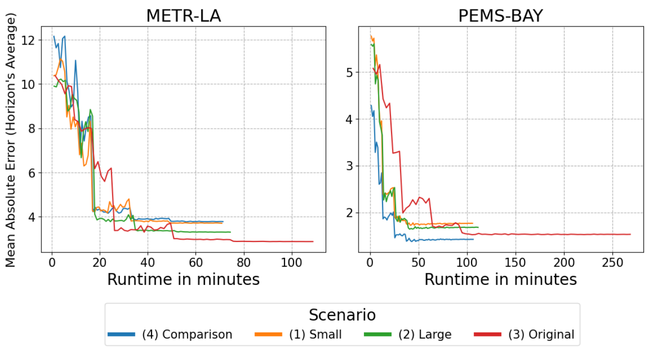


Figure 6: Mean Absolute Error measured after every epoch during training for each scenario (Experiments 1, 3).

Table 3: Number of missing values in the training set of each scenario for Experiment 2.

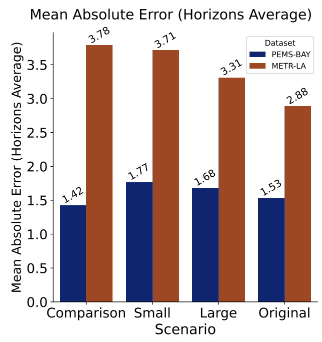
Scenario	Proportion (%)	Missing Values %	
		METR-LA	PEMS-BAY
(1) Small	100	5.93	0.0015
(2) Large	25	6.11	0.0017
	50	5.93	0.0018
	75	5.87	0.0015
	100	7.41	0.0017
(3) Original	100	7.13	0.0015
(4) Comparison	100	6.83	0.0007

having on average more neighbours (Figure 7d), and thus more information to use. This is consistent with results from another study [37], which found that increasing the number of sampled neighbours per node led to higher accuracy. A recent study demonstrates how training on large graphs improves accuracy by providing more information about the underlying spatial correlations [38]. These findings are also partially aligned with a large-scale GNN traffic forecasting study [31], where a large graph generally yielded lower MAE than two out of three of its subgraphs. Conversely, another study [33] using the DCRNN model, found lower MAE in a 20-sensor subgraph of METR-LA than in the original dataset, attributing it to improved spatial correlation capture in smaller, less heterogeneous networks. Therefore, this accuracy trend may not always apply to other models.

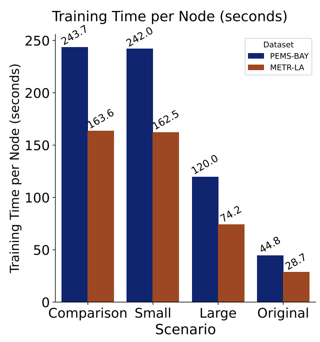
This experiment also highlights the better efficiency of larger graphs in terms of GPU usage and training time per node (Figures 7c, 7b). As the node count increases, so does the computational load due to increased node interactions and larger adjacency matrices [3]; operations which the D2STGNN model parallelizes, resulting in lower training time per node. Similar observations on increasing memory consumption have been made in another STGNN traffic forecasting study [39].

Scaling Complexity

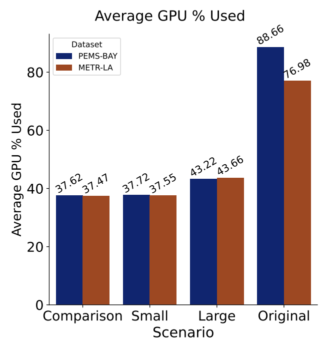
Experiment 2 presents mixed results for different datasets. For METR-LA, higher proportions result in lower MAE, but in PEMS-BAY, the opposite is true. For simpler road struc-



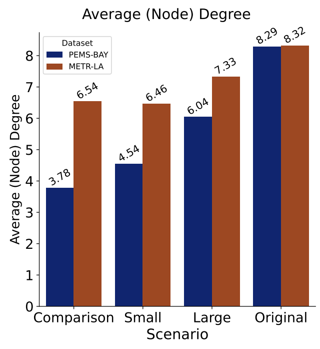
(a)



(b)



(c)



(d)

Figure 7: Mean Absolute Error, Training Time per Node in seconds, Average GPU % Used statistics, and Average (Node) Degree in each scenario (Experiments 1, 3).

tures and smaller datasets like METR-LA Scenario 2 (two intersections), increasing node count may be beneficial. However, in more intricate scenarios such as PEMS-BAY Scenario 2 (three intersections, longer time span), it could overwhelm the model, leading to lower accuracy. Notably, the standard deviation for MAE values was low (0.06 for METR-LA and 0.08 for PEMS-BAY). In another study [37], adding more edges to the graph was shown to improve accuracy, though with increasingly diminishing returns. This could explain the small differences observed in this experiment. This experiment's results also align with Experiment 1's findings on GPU utilization (Figures 9a, 7c, 10c); larger node count in a graph results in higher GPU usage.

Geographic Location Effect

In Experiment 3, PEMS-BAY Scenario 1, covering three intersections, shows lower accuracy than Scenario 4, which covers only one intersection. Similarly, in METR-LA, the more complex Scenario 4 shows lower accuracy than the simpler Scenario 1. A related study [33] using DCRNN on METR-LA, similarly observed this impact, though with findings showing higher accuracy for the more complex scenario. This could be due to modelling differences, the selection of sensors or varying missing value ratios. While the data is insufficient to draw definitive conclusions on whether more complex road structures consistently yield worse accuracy, these findings still highlight the influence of geographical location on model accuracy.

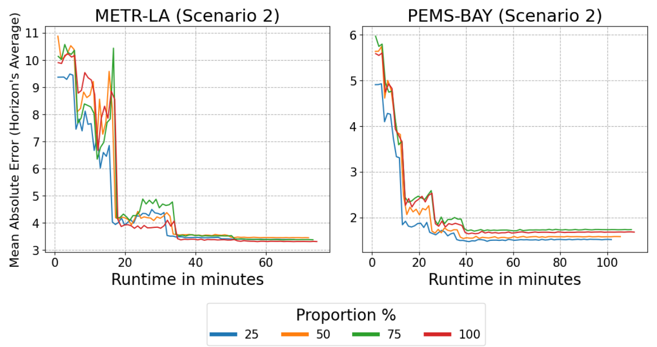
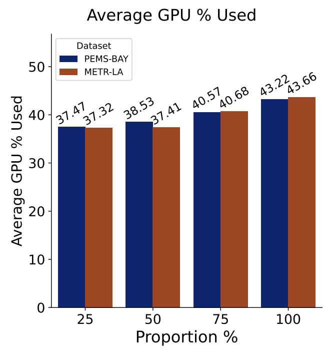
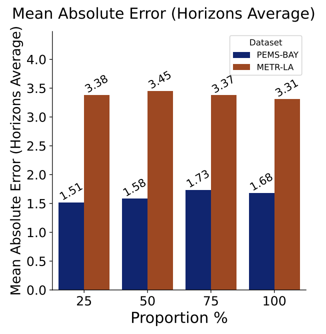


Figure 8: Mean Absolute Error measured after every epoch during training by proportion for Scenario 2 (Experiment 2).



(a)



(b)

Figure 9: Average GPU % used during training and mean absolute error by proportions of Scenario 2 (Experiment 2).

Furthermore, this experiment on METR-LA supports Experiment 1's results regarding larger graphs achieving better accuracy, as Scenario 4 shows inferior accuracy compared to Scenario 3. However, in PEMS-BAY, there is an observed improvement in MAE when the ratio of the missing values is lower. A related study [36], suggests that data completeness influences model performance. Finally, the computational requirements required for Scenario 4 in both datasets further support the trends observed in Experiments 1 and 2 and closely resemble those of Scenario 1 (see Figures 7c, 7b).

5.2 Verification of Results

To further verify this study's results, Experiments 1, 2 were expanded upon as detailed in Appendix B. Experiment 1's extension involved sampling increasingly larger subsets between Scenarios 2 and 3, which confirmed that the MAE generally decreases for both datasets (Figure 10a). However, after about 100 sensors, the MAE remained relatively stable. For Experiment 2, where sensors were sampled in maximally scattered fashion with 10% increments, the MAE showed stability with a low standard deviation of 0.068 for METR-LA, and 0.025 for PEMS-BAY.

Figure 10d generally verifies the increase in average node degree observed in Experiment 1. This increase can be attributed to the growing number of edges as more nodes are added, leading to a denser network. The average node de-

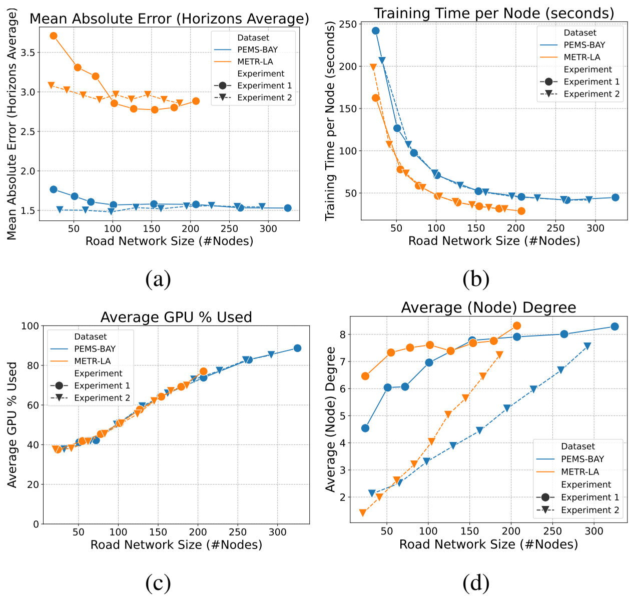


Figure 10: Mean Absolute Error, Training Time per Node in seconds, Average GPU % Used statistics, and Average (Node) Degree for verifying Experiments 1, 2.

gree eventually somewhat stabilizes, and the trends for each dataset could be linked to the way the subsets grow (see Appendix B); when denser areas are added, the node degree tends to increase. Notably, the increase in average node degree appears to correspond with the decrease in MAE, similarly to observations in another study [37]; however, the evidence is insufficient to draw a definitive conclusion. For Experiment 2, the growth is linear, corresponding to a proportional increase in network density. However, despite the linear growth in neighbours, the MAE remains stable. This might be because denser placement of sensors on highways could potentially offer limited additional information. Thus, increasing the density of maximally scattered sensors raises the average node degree, but does not seem to significantly impact accuracy.

Additionally, Figures 10c, 10b confirm the observed efficiency trends of the experiments and provide additional insights. GPU usage increases linearly with the number of nodes. Around 325 nodes, as the GPU approaches its limits, training time per node slightly increases. Despite being a high-accuracy model, D2STGNN has previously struggled to scale to significantly larger datasets (2×10^3 nodes) due to its complexity [31; 38].

5.3 Limitations

Limitations of this study should be acknowledged to understand the context and applicability of the findings. First, while using two datasets provides a broader perspective, both are confined to California. This geographic limitation may restrict the applicability of the results to other regions with different traffic patterns and infrastructure. Second, the experiments were conducted on a smaller scale, providing initial insights. Ideally, larger datasets should be used, and experiments should be repeated to minimize the potential impacts of concurrent activities on hardware resources. The limited availability of such datasets and the computational resources required for larger datasets, given the time constraints of this study, made fully addressing these limitations unfeasible. Third, broken sensors, which can affect the results, were partially addressed. With more time, advanced techniques [36] could be applied to mitigate their impact more effectively. Fourth, biases introduced by sensor selection were accounted for in Experiment 1 using Experiment 3, and partially mitigated in Experiment 2 using Algorithm 1 to ensure maximal data representation. Algorithm 1 however, may not always produce the optimal solution. Lastly, while the results from the D2STGNN may generalize to other GNNs, they cannot fully generalize to all GNNs as discussed in 5.1.

6 Responsible Research

This section addresses the ethical considerations of this research and discusses the steps taken to ensure the reproducibility of the results.

6.1 Ethical Aspects

Ethical issues related to research advances in traffic forecasting need to be considered. Primarily, data privacy is a critical concern as traffic forecasting systems gain popularity, the volume of collected data will increase, leading to greater risks of data breaches and misuse. The datasets used in this experiment are anonymous and publicly available to mitigate such risks.

Another significant ethical issue is the potential for increased surveillance. Detailed traffic data collection can lead to environments where individual movements are closely monitored. Experiment 2 demonstrates that increasing the number of sensors in a specific area does not significantly enhance accuracy, and may even diminish it. The hope is that this finding discourages governments from installing unnecessary sensors, thus protecting individual privacy rights.

Additionally, the benefits of traffic forecasting systems may not be equitably distributed, potentially exacerbating socioeconomic disparities. Wealthier areas might experience more significant improvements due to better infrastructure, while lower-income regions may continue to suffer from poor traffic conditions. Research into the scalability of traffic forecasting GNNs aims to reduce the resources needed for such systems, eventually making traffic forecasting more accessible to lower-income areas.

Traffic forecasting also has a positive impact on the environment by helping to reduce emissions and improve air quality. While the effect may be smaller, reducing training requirements for GNNs also contributes to environmental goals. Ultimately, traffic forecasting aims to improve the everyday lives of people, and conducting research to enhance it while considering ethical implications is beneficial.

6.2 Reproducibility of Methods

Reproducibility is key in responsible research. For this study, a GitHub repository [40] which includes the implementation of Algorithm 1, code to generate the sensor subsets, visualizations, and metrics shown in this paper, as well as instructions on running the experiments presented, has been made publicly available. In this repository, the indices and IDs of the sensor subsets are also included. The datasets [41] and source code for the model [35] are also publicly available. The provided model sets a random seed to ensure the reproducibility of the results. However, efficiency metrics can still be biased due to concurrent activities utilizing hardware resources. To minimize their impact on the results, only the GPU usage of the specific process is measured.

7 Conclusions and Future Work

This study evaluated the scalability of the D2STGNN model for traffic forecasting using two real-world datasets, focusing on how varying map sizes, graph complexities, and geographic locations affect model accuracy and computational requirements.

Findings. The findings from Experiment 1, reveal that larger graphs, result in shorter training times per node, provided the GPU is not operating at maximum capacity, and parallelization is implemented, as the efficiency is achieved through better GPU utilization. Additionally, nodes in larger graphs have a greater degree, which could contribute to higher accuracy [37]. In Experiment 2, the model showed robustness in accuracy when increasing graph complexity, while GPU usage increased, verifying a direct relationship between node count and computational demands. Lastly, Experiment 3 showed that the geographic location of the sensors can impact accuracy.

Future Work. Proposals for future work include (i) *Using larger, more diverse datasets, and repeating the experiments*. This would address the limitations of this study's geographical scope, and minimize the potential impacts of concurrent activities on hardware resources, allowing for more generalizable conclusions. (ii) *Evaluating different dimensions of scalability,* such as increasing feature count or having more temporal data, to examine if similar patterns occur. (iii) *Investigating methods to make dynamic graph learning modules more scalable*. Models that consider the dynamic characteristics of traffic networks show exceptional accuracy performance. However, their complexity prevents them from scaling to large datasets [31]. Research into enhancing their scalability can potentially help keep their accuracy benefits while maintaining practical usability.

References

[1] J. Wootton, A. Garcia-Ortiz, and S. Amin, "Intelligent transportation systems: a global perspective," *Mathematical and computer modelling*, vol. 22, no. 4-7, pp. 259–268, 1995.

[2] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[3] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert Systems with Applications*, vol. 207, p. 117921, Nov. 2022. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2022.117921

[4] Z. Lu, W. Lv, Z. Xie, B. Du, and R. Huang, "Leveraging graph neural network with lstm for traffic speed prediction," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2019, pp. 74–81.

[5] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4189–4196.

[6] R. Dai, S. Xu, Q. Gu, C. Ji, and K. Liu, "Hybrid spatio-temporal graph convolutional network: Improving traffic prediction with navigation data," in *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining*, 2020, pp. 3074–3082.

[7] J. Rico, J. Barateiro, and A. Oliveira, "Graph neural networks for traffic forecasting," 2021.

[8] L. Wu, P. Cui, J. Pei, L. Zhao, and X. Guo, *Graph neural networks: foundation, frontiers and applications*. Singapore: Springer Nature Singapore, 2022, ch. 6, pp. 101–119.

[9] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. ACM, Jul. 2019. [Online]. Available: http://dx.doi.org/10.1145/3292500.3330925

[10] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," 2020.

[11] M. Fey, J. E. Lenssen, F. Weichert, and J. Leskovec, "Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings," 2021.

[12] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou, and Y. Dai, "Towards efficient large-scale graph neural network computing," 2018.

[13] W. Jiang, J. Luo, M. He, and W. Gu, "Graph neural network for traffic forecasting: The research progress," *ISPRS International Journal of Geo-Information*, vol. 12, no. 3, p. 100, 2023.

[14] B. L. Smith and M. J. Demetsky, "Traffic flow forecasting: comparison of modeling approaches," *Journal of transportation engineering*, vol. 123, no. 4, pp. 261–266, 1997.

[15] B. M. Williams, P. K. Durvasula, and D. E. Brown, "Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models," *Transportation Research Record*, vol. 1644, no. 1, pp. 132–141, 1998.

[16] M. S. Ahmed and A. R. Cook, "Analysis of freeway traffic time-series data by using box-jenkins techniques," *Transportation Research Record*, no. 722, 1979.

[17] L. Zhang, Q. Liu, W. Yang, N. Wei, and D. Dong, "An improved k-nearest neighbor model for short-term traffic flow prediction," *Procedia-Social and Behavioral Sciences*, vol. 96, pp. 653–662, 2013.

[18] Y. Liu and H. Wu, "Prediction of road traffic congestion based on random forest," in *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2, 2017, pp. 361–364.

[19] G. Aifadopoulou, C. Bratsas, K. Koupidis, A. Chatzopoulou, J.-M. Salanova, and P. Tzenos, "Short-term prediction of the traffic status in urban places using neural network models," in *Data Analytics: Paving the Way to Sustainable Urban Mobility: Proceedings of 4th Conference on Sustainable Urban Mobility (CSUM2018), 24-25 May, Skiathos Island, Greece*. Springer, 2019, pp. 181–188.

[20] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[21] R. Toncharoen and M. Piantanakulchai, "Traffic state prediction using convolutional neural network," in *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2018, pp. 1–6.

[22] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," 2019.

[23] A. Belhadi, Y. Djenouri, D. Djenouri, and J. C.-W. Lin, "A recurrent neural network for urban long-term traffic flow forecasting," *Applied Intelligence*, vol. 50, pp. 3252–3265, 2020.

[24] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," 2022.

[25] X. Liu, Y. Liang, C. Huang, H. Hu, Y. Cao, B. Hooi, and R. Zimmermann, "Do we really need graph neural networks for traffic forecasting?" 2023.

[26] A. Cini, I. Marisca, D. Zambon, and C. Alippi, "Graph deep learning for time series forecasting," 2023.

[27] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," 2020.

[28] A. Gandhi, Aakankasha, S. Kaveri, and V. Chaoji, "Spatio-temporal multi-graph networks for demand forecasting in online marketplaces," in *ECML-PKDD 2021*, 2021. [Online]. Available: https://www.amazon.science/publications/spatio-temporal-multi-graph-networks-for-demand-forecasting-in-online-marketplaces

[29] F. Frasca, E. Rossi, D. Eynard, B. Chamberlain, M. Bronstein, and F. Monti, "Sign: Scalable inception graph neural networks," 2020.

[30] A. Cini, I. Marisca, F. M. Bianchi, and C. Alippi, "Scalable spatiotemporal graph neural networks," *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, 2023.

[31] X. Liu, Y. Xia, Y. Liang, J. Hu, Y. Wang, L. Bai, C. Huang, Z. Liu, B. Hooi, and R. Zimmermann, "Largest: A benchmark dataset for large-scale traffic forecasting," 2023.

[32] Z. Shao, Z. Zhang, W. Wei, F. Wang, Y. Xu, X. Cao, and C. S. Jensen, "Decoupled dynamic spatial-temporal graph neural network for traffic forecasting," *arXiv preprint arXiv:2206.09112*, 2022.

[33] S. Rahmani, "Revisiting DCRNN: Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," Apr 2023. [Online]. Available: https://medium.com/@saeedrmd/revisiting-dcrnn-diffusion-convolutional-recurrent-neural-network-data-driven-traffic-forecasting-caeecbe3281b

[34] M. Lablack and Y. Shen, "Spatio-temporal graph mixformer for traffic forecasting," *Expert Systems with Applications*, vol. 228, p. 120281, 2023.

[35] Z. Shao, Z. Zhang, W. Wei, F. Wang, Y. Xu, X. Cao, and C. S. Jensen. Decoupled dynamic spatial-temporal graph neural network for traffic forecasting. [Online]. Available: https://github.com/zezhishao/D2STGNN

[36] J. Zuo, K. Zeitouni, Y. Taher, and S. Garcia-Rodriguez, "Graph convolutional networks for traffic forecasting with missing values," *Data Mining and Knowledge Discovery*, vol. 37, no. 2, pp. 913–947, 2023.

[37] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018.

[38] Y. Jiang, X. Li, Y. Chen, S. Liu, W. Kong, A. F. Lentzakis, and G. Cong, "Sagdfn: A scalable adaptive graph diffusion forecasting network for multivariate time series forecasting," *arXiv preprint arXiv:2406.12282*, 2024.

[39] J. Han, W. Zhang, H. Liu, T. Tao, N. Tan, and H. Xiong, "Bigst: Linear complexity spatio-temporal graph neural network for traffic forecasting on large-scale road networks," *Proceedings of the VLDB Endowment*, vol. 17, no. 5, pp. 1081–1090, 2024.

[40] Danae Savvidi. Data processing repository for bachelor thesis. [Online]. Available: https://github.com/danaesav/RP-Data-Preprocessing

[41] R. Jiang, D. Yin, Z. Wang, Y. Wang, J. Deng, H. Liu, Z. Cai, J. Deng, X. Song, and R. Shibasaki. Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction. [Online]. Available: https://github.com/deepkashiwa20/DL-Traff-Graph

A Detailed Speed Predictions of Two Sensors (Experiment 1)

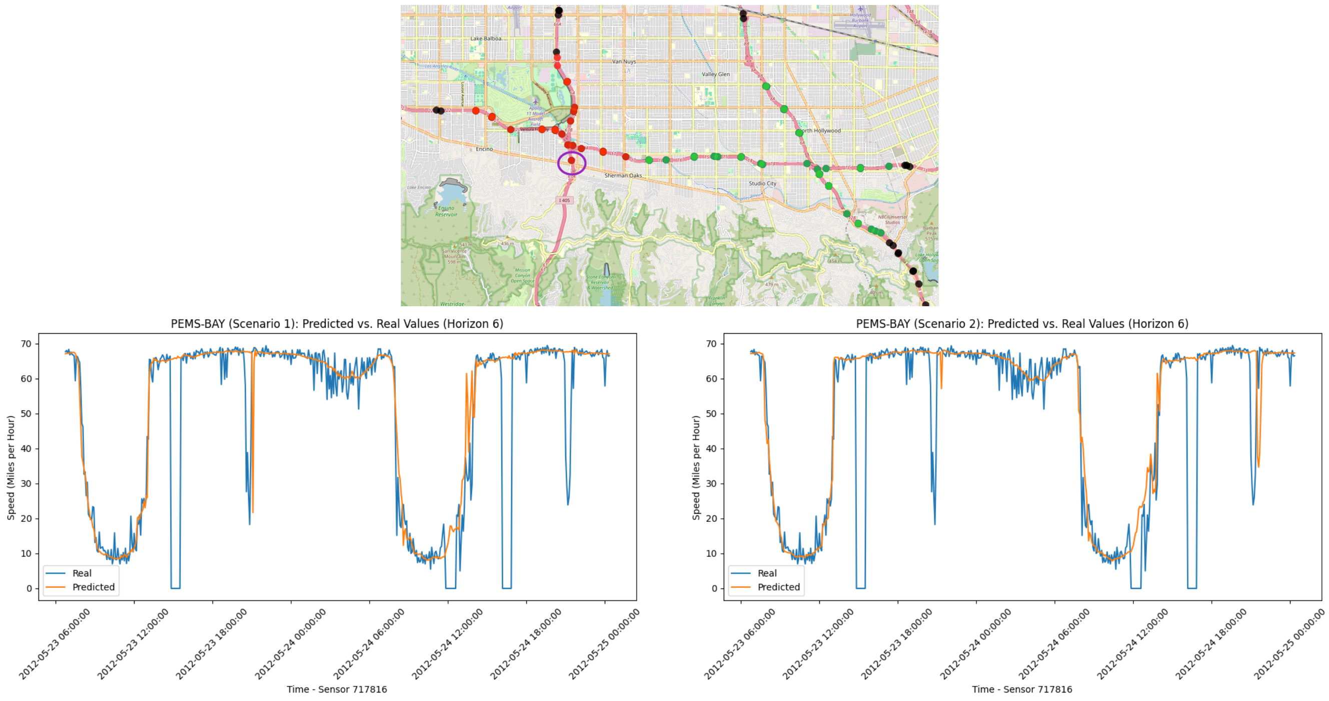


Figure 11: Real vs Predicted speed values for Sensor 717816 in scenarios 1, 2 for the METR-LA dataset.

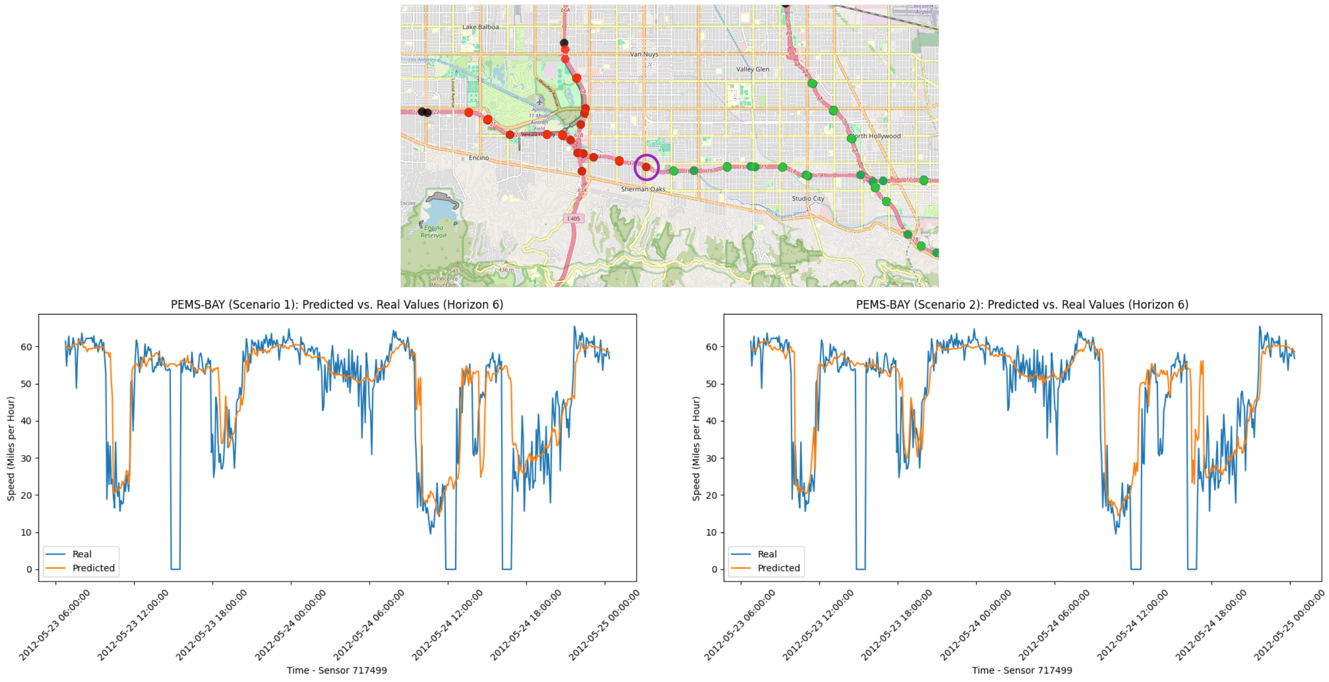
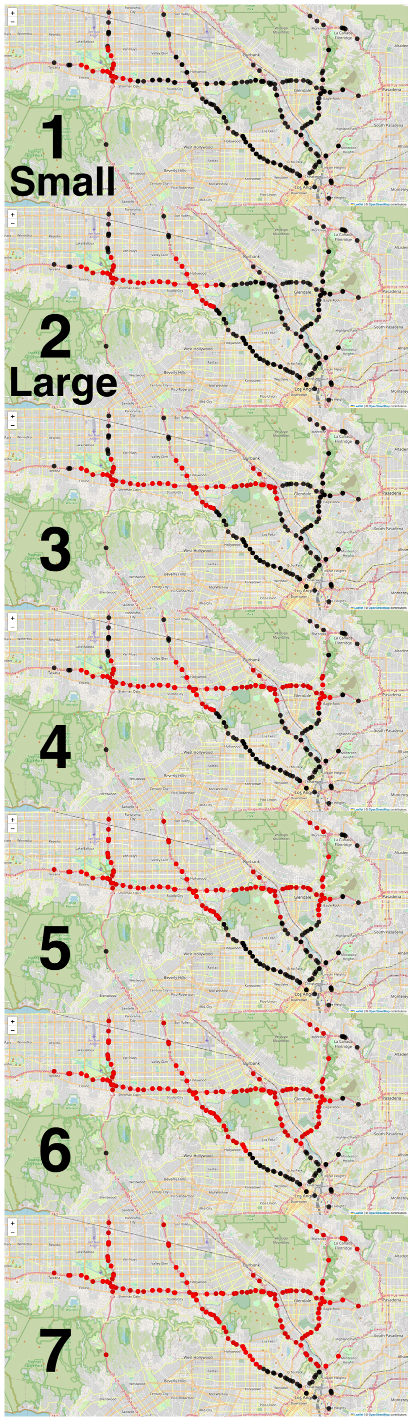


Figure 12: Real vs Predicted speed values for Sensor 717499 in scenarios 1, 2 for the METR-LA dataset.

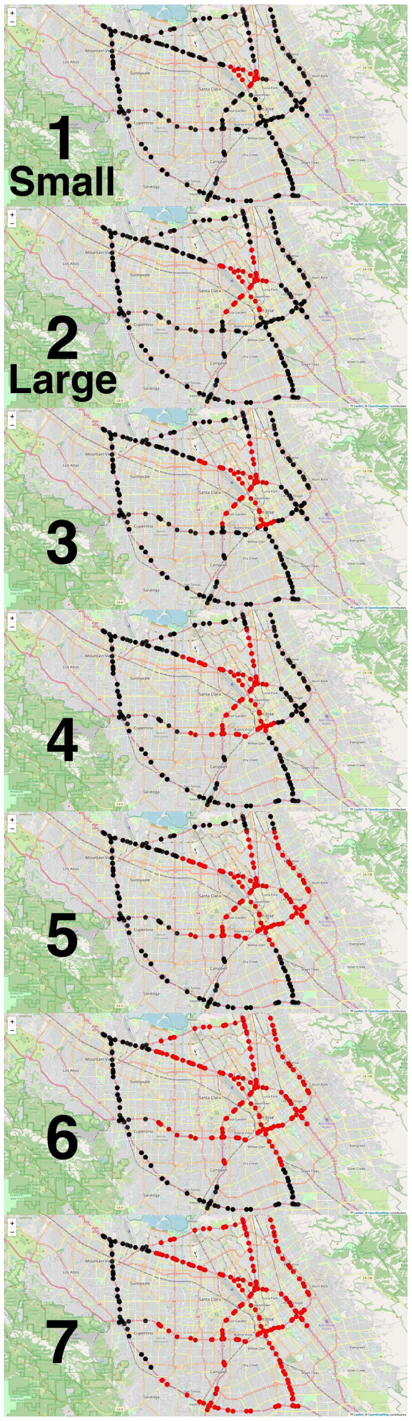
B Verification of Results

B.1 Experiment 1

Figure 13 shows the subsets used to verify the results of Experiment 1. The initial small and large subsets are used, as well as the whole dataset. For METR-LA the scaling happens first along the x-axis and then along the y-axis to cover more parts of the dataset. In PEMS-BAY, the expansion occurs generally along both axes. The exact number of sensors, along with their missing values percentages, and MAE values are presented in Table 4.



(a) METR-LA



(b) PEMS-BAY

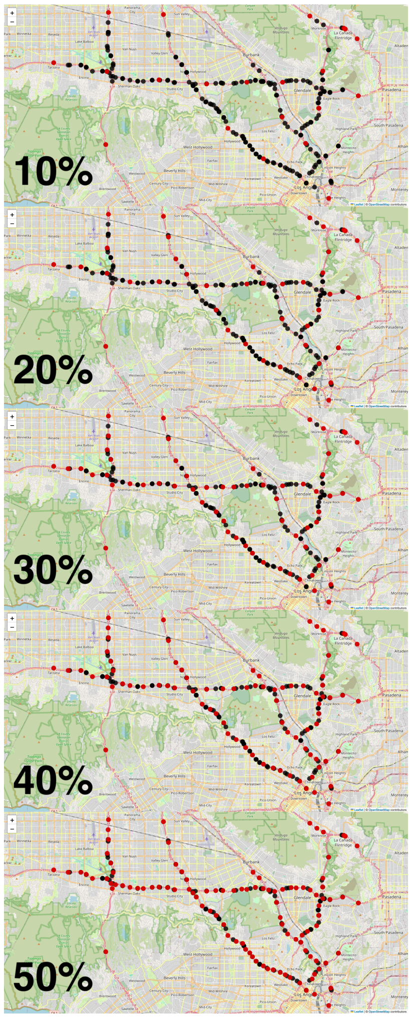
Figure 13: Subsets taken to verify the results of Experiment 1.

Table 4: Number of sensors sampled, percentage of missing values in the training set, and mean absolute error for each subset.

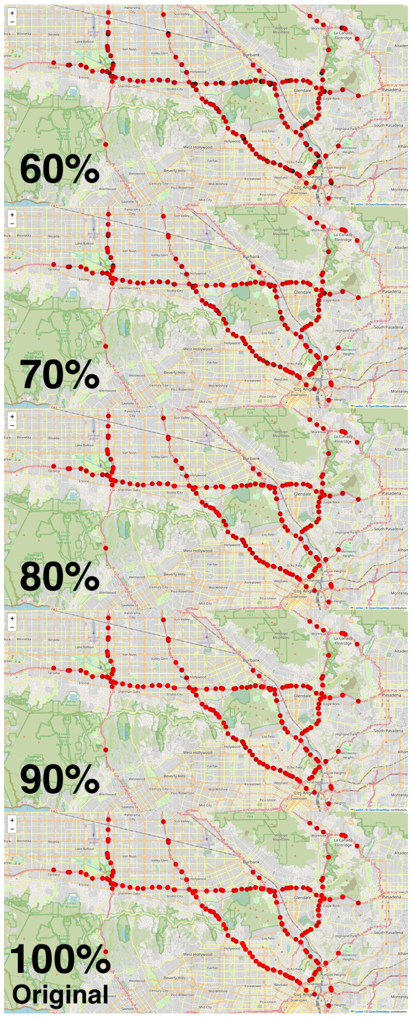
Subset	Number of Sensors		Missing Values %		Mean Absolute Error	
	METR-LA	PEMS-BAY	METR-LA	PEMS-BAY	METR-LA	PEMS-BAY
Subset 1 (Small)	24	24	5.93	0.0015	3.71	1.77
Subset 2 (Large)	55	51	7.41	0.0017	3.31	1.68
Subset 3	78	72	6.71	0.0016	3.2	1.61
Subset 4	102	101	7.41	0.0017	2.86	1.57
Subset 5	127	153	7.4	0.0015	2.79	1.58
Subset 6	154	207	7.36	0.0015	2.77	1.58
Subset 7	179	264	7.2	0.0015	2.8	1.53
Original	207	325	7.13	0.0015	2.88	1.52

B.2 Experiment 2

Figures 14, 15 illustrates the sensors sampled from the original datasets with varying proportions, to verify the results of Experiment 2. Details on the number of sensors, missing values % in the training sets and the mean absolute errors are shown in Table 5.

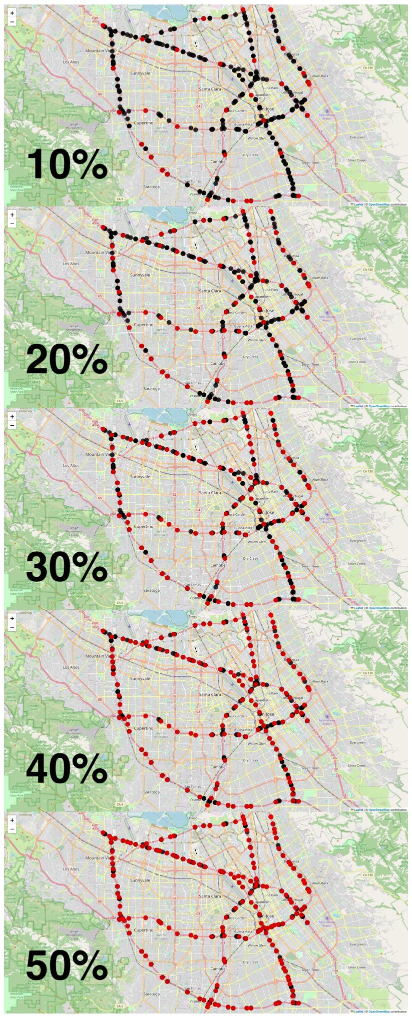


(a) METR-LA proportions 10-50%

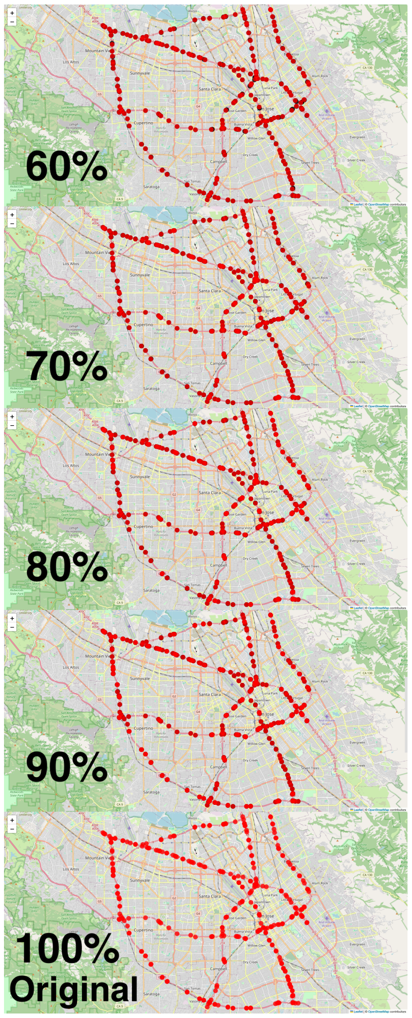


(b) METR-LA proportions 60-100%

Figure 14: METR-LA: Subsets taken to verify the results of Experiment 2.



(a) PEMS-BAY proportions 10-50%



(b) PEMS-BAY proportions 60-100%

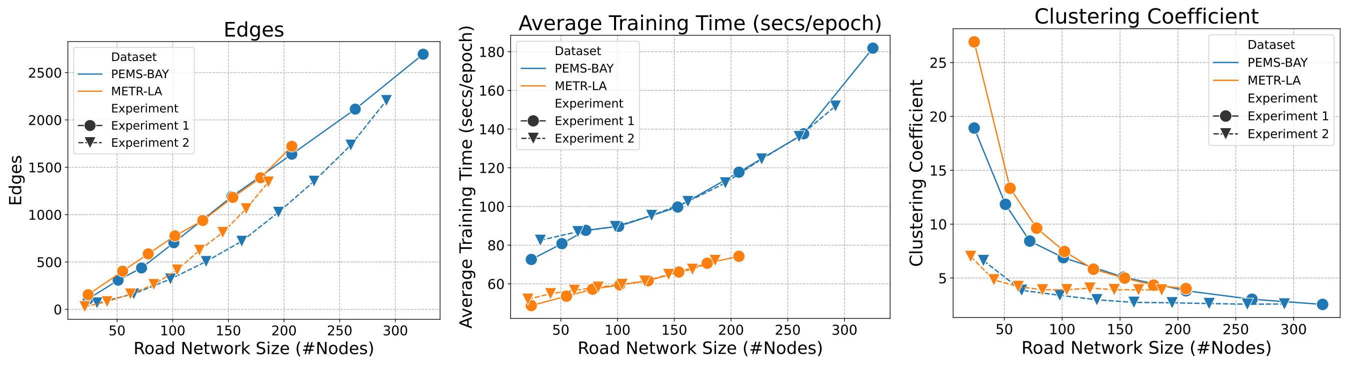
Figure 15: PEMS-BAY: Subsets taken to verify the results of Experiment 2.

Table 5: Number of sensors sampled, percentage of missing values in the training set, and mean absolute error for each subset in Experiment 2.

Scenario	**Proportion (%)**	**Number of Sensors**		**Missing Values %**		**Mean Absolute Error**	
		METR-LA	**PEMS-BAY**	**METR-LA**	**PEMS-BAY**	**METR-LA**	**PEMS-BAY**
(2) Large	25	14	13	6.11	0.0017	3.38	1.51
	50	28	26	5.93	0.0018	3.45	1.58
	75	42	39	5.87	0.0015	3.37	1.73
	100	55	51	7.41	0.0017	3.31	1.68
(3) Original	10	21	32	7.13	0.0015	3.08	1.50
	20	41	65	7.13	0.0015	3.02	1.50
	30	62	98	7.13	0.0015	2.96	1.48
	40	83	130	7.13	0.0015	2.90	1.54
	50	104	162	7.13	0.0015	2.97	1.52
	60	124	195	7.13	0.0015	2.90	1.55
	70	145	227	7.13	0.0015	2.96	1.56
	80	166	260	7.13	0.0015	2.90	1.55
	90	186	292	7.13	0.0015	2.86	1.54
	100	207	325	7.13	0.0015	2.88	1.53

B.3 Additional Results

The number of edges (Figure 16a) grows linearly with the number of nodes as the network size grows in Experiment 1, while it grows slightly exponentially in Experiment 2. The average training time in seconds per epoch (Figure 16b) also initially grows somewhat linearly, with a faster increase at around 325 nodes, which corresponds with the GPU reaching its limits. Clustering coefficient (Figure 16c) is a measure of the degree to which nodes in a graph tend to cluster together[3] (i.e. graph connectivity). It decreases as the network size grows in Experiment 1. In Experiment 2, it slightly decreases and then remains stable.



(a) (b) (c)

Figure 16: Additional results from verifying Experiments 1, 2.

[3]https://en.wikipedia.org/wiki/Clustering_coefficient

C Decoupled Dynamic Spatio-Temporal Graph Neural Network

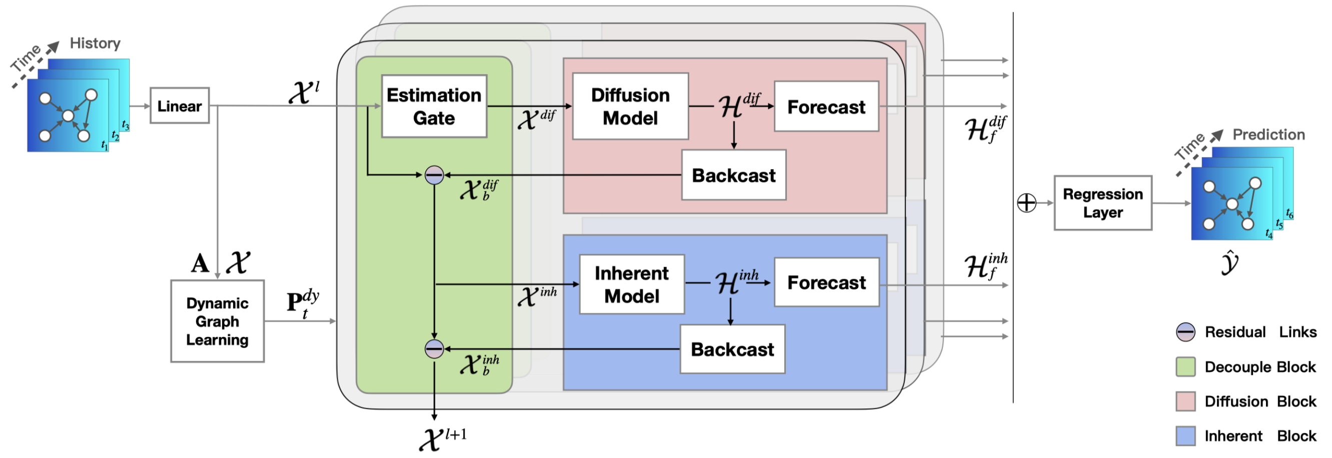
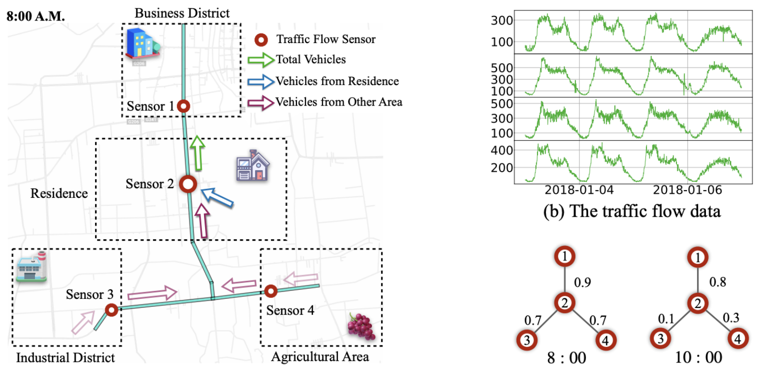


Figure 17: The overall architecture of the proposed D2STGNN. The decouple block (green) decomposes each time series in traffic signals into two hidden time series, which are subsequently handled by the diffusion block (pink) and inherent block (blue). Moreover, the dynamic graph learning module generates dynamic spatial dependency for the diffusion model. Taken from [32].



(a) An example of traffic flow system in the 8:00 a.m.

(b) Dynamic spatial dependency.

Figure 18: An example of the traffic flow system. Taken from [32].

The following description is modified from [32]. Figure 18 illustrates a traffic flow system with important locations monitored by sensors that record vehicle counts over time. Each sensor's data is influenced by two factors: a diffusion signal and a non-diffusion signal. For instance, vehicles passing Sensor 2 (green arrow) at 8 a.m. originate from two sources shown in Figure 18a: directly nearby (blue arrow) and from adjacent areas (wine-red arrow), such as the industrial district (Sensor 3) and agricultural area (Sensor 4). The former is independent of other sensors, while the latter is influenced by diffusion processes. These are referred to as hidden inherent time series and hidden diffusion time series, respectively. Furthermore, the spatial dependencies within the road network are dynamic. For example, as depicted in Figure 18b, at 8 a.m., the traffic at Sensors 3 and 4 significantly affects Sensor 2, whereas by 10 a.m., the influence diminishes.