Traffic Flow Forecasting on Spatio-Temporal Data

Daniel Ochana Michal Maya
Supervisor: Ya-Wei Eileen Lin

April 2025

1 Abstract

Traffic flow forecasting is a crucial task for urban mobility and intelligent transportation systems. In this work, we explore graph-based deep learning approaches to model traffic dynamics across space and time using real-world datasets from multiple cities. We represent traffic data as weighted directed graphs, where nodes correspond to spatial locations and edges represent road connections, weighted by the average driving speed in different time slots.

We examine different methods, including MagNet Graph Neural Networks (GNNs) [1] and dynamic graph models such as DyRep [2] and Self-Supervised Learning (ST-SSL) [3]. Our primary focus is on adapting MagNet [1] to effectively handle weighted spatial-temporal information, evaluating symmetric and non-symmetric adjacency representations. We present our methodology, dataset processing pipeline, and experimental results, demonstrating the potential of graph-based methods for traffic flow forecasting.

2 Introduction

Traffic flow forecasting is a crucial task in intelligent transportation systems, aiding in congestion management, route optimization, and urban planning. Traditional forecasting methods, such as time-series models and statistical regression techniques, often fail to capture the intricate spatial and temporal dependencies present in real-world transportation networks. Graph-based deep learning approaches, particularly Graph Neural Networks (GNNs), have gained significant attention for their ability to model these dependencies, leveraging the inherent graph structure of road networks.

In this work, we investigate various graph-based approaches for traffic flow forecasting using multiple real-world datasets, including NYC Taxi [4], PeMSD8 [5], and METR-LA [6]. These datasets provide different traffic dynamics, covering ride-sharing services, highway sensor data, and urban traffic monitoring. We

represent the data as a series of weighted directed graphs, where nodes correspond to spatial locations, and edges represent road connections weighted by the average driving speed within specific time slots. Each dataset was preprocessed to construct suitable graph representations and feature matrices, enabling a comparative analysis of different modeling strategies.

We experimented with several methods to tackle this problem, including MagNet Graph Neural Networks (GNNs) [1], DyRep for dynamic graphs [2], and self-supervised learning (ST-SSL) [3] techniques. Ultimately, we based our prediction model on MagNet [1], an advanced GNN that incorporates directional and magnitude information using a magnetic Laplacian matrix. Since MagNet [1] has not been previously applied to weighted directed graphs, we introduced two adaptations: a symmetric version, where directional information is incorporated only in the phase matrix, and a non-symmetric version, where both the phase matrix and the adjacency matrix retain weighted directional information.

Our dataset processing pipeline involved extensive experimentation with different labeling strategies and hyperparameter tuning to optimize performance. The results demonstrated that the symmetric adaptation of MagNet [1] achieved superior performance, with test accuracies of 86.0% for "into traffic" and 85.9% for "from traffic", compared to the non-symmetric model's 84.8% accuracy for "into traffic". These findings highlight the potential of graph-based deep learning for traffic forecasting and suggest promising directions for future research, including incorporating temporal edges to construct a more comprehensive spatio-temporal graph representation and further exploring dynamic graph models such as DyRep [2] and self-supervised learning frameworks.

3 Background

This section provides an overview of key concepts relevant to our work, including graph-based machine learning, traffic flow forecasting, and prior research on dynamic and spatial-temporal graph models.

3.1 Dynamic Graphs

A dynamic graph is represented as a time-dependent graph:

$$G_t = (V_t, E_t, X_t, S_t) \tag{1}$$

where:

- V_t is the set of nodes (vertices) at time t.
- E_t is the set of edges (connections between nodes) at time t.
- $X_t \in \mathbb{R}^{|V_t| \times F}$ is the feature matrix at time t, where each node has a F-dimensional feature vector.
- $S_t \in \mathbb{R}^{|V_t| \times |V_t|}$ is the dynamically learned adjacency matrix that represents the relationships between nodes.

To capture spatio-temporal dependencies, a generalized graph convolution operation is applied:

$$H^{(l+1)} = \sigma(S_t H^{(l)} W^{(l)}) \tag{2}$$

where:

- $H^{(l)} \in \mathbb{R}^{|V_t| \times d_l}$ is the node embedding matrix at layer l.
- $W^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is the trainable weight matrix.
- S_t captures both predefined and learned graph structures.
- $\sigma(\cdot)$ is a non-linear activation function.

3.2 Magnetic Laplacian

Graphs can be described by an adjacency matrix:

$$A(u,v) = \begin{cases} 1, (u,v) \in E \\ 0, \text{otherwise} \end{cases} \tag{3}$$

For undirected graphs, we define the Laplacian matrix:

$$L = D - A \tag{4}$$

where D is a diagonal degree matrix. When implemented in a directed graph, L is not symmetric, and therefore, its eigenvalues become complex.

For the presence of an edge, we use the symmetrized adjacency matrix:

$$A_s(u,v) = \frac{1}{2}\left(A(u,v) + A(v,u)\right) \tag{5}$$

Along with the corresponding degree matrix:

$$D_s(u,u) = \sum_v A_s(u,v) \tag{6}$$

For the directional information, we use the phase matrix:

$$\Theta^q(u,v) = 2\pi q \left(A(u,v) - A(v,u)\right) \tag{7}$$

Finally, we define the Hermitian adjacency matrix:

$$H^q = A_s \cdot \exp(i\Theta^q) \tag{8}$$

The full information will be presented in the Magnetic Laplacian matrix:

$$L^q = D_s - H^q = D_s - A_s \cdot \exp(i\Theta^q) \tag{9}$$

Since this matrix is positive semi-definite (PSD), the eigenvalues are real and nonnegative.

The Laplacian can also be described as:

$$L = U \Lambda U^\dagger \tag{10}$$

Where:

- U is the eigenvector matrix.
- Λ is the diagonal matrix of eigenvalues.

The smaller eigenvalues correspond to the lower frequencies in graphs.

3.3 Self-supervised learning

Self-supervised learning (SSL) is a paradigm where a model learns representations from unlabeled data by defining surrogate tasks known as pretext tasks. These tasks create pseudo-labels from the raw data, allowing the model to learn general features that can be used for downstream tasks like classification or forecasting.

Objective Function in SSL Given a dataset $X = \{x_1, x_2, ..., x_N\}$, self-supervised learning aims to learn a representation function $f_\theta(\cdot)$ by optimizing an auxiliary loss function:

$$\theta^* = \arg\min_\theta \mathcal{L}_{SSL}(f_\theta(X)) \tag{11}$$

where $\mathcal{L}_{SSL}$ is the self-supervised loss designed to capture meaningful structures in the data.

Contrastive Learning for Graphs One common SSL approach is contrastive learning, which learns by maximizing agreement between positive pairs (x, x^+) while minimizing agreement with negative samples (x, x^-). The contrastive loss, based on the InfoNCE (Noise Contrastive Estimation), is given by:

$$\mathcal{L}_{contrast} = -\log \frac{\exp(\text{sim}(f_\theta(x), f_\theta(x^+))/\tau)}{\sum_{x^-} \exp(\text{sim}(f_\theta(x), f_\theta(x^-))/\tau)} \tag{12}$$

where:

- $\text{sim}(a, b)$ measures similarity (e.g., cosine similarity $\frac{a \cdot b}{\|a\|\|b\|}$),
- τ is a temperature scaling parameter,
- x^+ is an augmented view (positive sample) of x,
- x^- are negative samples from other nodes.

4 Methods

In this section, we describe our approach to representing traffic data as graphs, preprocessing techniques, and the models we employed, including MagNet [1], DyRep [2], and self-supervised learning methods [3].

4.1 Dyrep

The first method we tried was based on the method presented in the paper "DYREP: Learning Representations over Dynamic Graphs" [2]. In this paper, the authors present a new approach for understanding and representing data in graphs that change over time. The framework learns from dynamic graphs that evolve based on two key processes:

- **Topological Evolution (Association Process)**: The graph structure changes over time, meaning V_t and E_t can be updated dynamically.
- **Node Interactions (Communication Process)**: Nodes interact dynamically, leading to time-dependent updates in X_t and S_t.

DyRep [2] uses a two-time scale model to capture these changes:

A temporal-attentive representation network to update node representations based on the evolving graph structure. A deep temporal point process to model the timing of these changes, capturing the dependencies between topological evolution and node interactions. This evolution is modeled using a temporal point process, where the conditional intensity function for an event between nodes u and v at time t is given by:

$$\lambda_{u,v}(t) = f_k(g_k(z_u(t), z_v(t))) \tag{13}$$

where:

- $z_u(t), z_v(t)$ are the node embeddings at time t.
- $g_k(\cdot)$ models the compatibility between nodes.
- $f_k(\cdot)$ ensures non-negative intensity values.

We aimed to represent our problem as a dynamic graph where the nodes correspond to static locations in the city, and the edges represent dynamic rides between those nodes. The edge weights correspond to the average speed at a given time. In our case, Topological Evolution was used only for changes in the structure of edges, while the nodes remained static over time.

Despite our efforts to adapt the DyRep framework [2] to suit our data and problem formulation, we concluded that it was not suitable for our needs and decided to explore alternative approaches.

4.2 Spatio-Temporal Self-Supervised Learning

The second method we explored is from the paper "Spatio-Temporal Self-Supervised Learning for Traffic Flow Prediction", which introduces Spatio-Temporal Self-Supervised Learning (ST-SSL) [3] to model both spatial and temporal variations in traffic data. The framework uses Structure Learning Convolution (SLC) to dynamically learn graph structures and Pseudo Three-Dimensional (P3D) Convolution to model temporal dependencies. Additionally, it applies

self-supervised learning (SSL) with auxiliary tasks and adaptive data augmentation to improve accuracy.

In this method, traffic data is represented as a dynamic graph, where:

- Nodes represent traffic sensors or locations.
- Edges represent road connectivity or learned relationships.
- Node features (e.g., speed, flow, density) evolve over time.

The model learns a forecasting function:

$$X_{t+Q} = h(X_{t-P}, ..., X_t, S_t; \theta) \tag{14}$$

where $h(\cdot)$ predicts future traffic states given past observations.

Self-supervised learning enhances node representations through:

- **Temporal SSL (Future Prediction):** The model predicts the future state X_{t+1} from past data:

$$\mathcal{L}_{temp} = \|X_{t+1} - \hat{X}_{t+1}\|_2^2 \tag{15}$$

- **Spatial SSL (Graph Structure Consistency):** A contrastive loss ensures embeddings remain stable under perturbations:

$$\mathcal{L}_{spatial} = \sum_{v \in V} \log \frac{\exp(\text{sim}(f_\theta(x_v), f_\theta(x_v^+)))}{\sum_{x_v^-} \exp(\text{sim}(f_\theta(x_v), f_\theta(x_v^-)))} \tag{16}$$

The final SSL objective combines both losses:

$$\mathcal{L}_{SSL} = \lambda_1 \mathcal{L}_{temp} + \lambda_2 \mathcal{L}_{spatial} \tag{17}$$

However, this representation was not ideal for our data, as we needed traffic features on edges rather than nodes, treating nodes as static spatial locations instead. This adjustment was necessary to better match our dataset's structure.

4.3 MagNet

We define the directed graph Fourier Transform for a signal x by:

$$\hat{x} = U^\dagger x \tag{18}$$

Now, we can define the spectral convolution matrix, used as a filter:

$$Y = U \Sigma U^\dagger \tag{19}$$

Where Σ is a diagonal matrix containing filter coefficients.

Now, we can obtain the ℓ^{th} layer forward function of the network:

$$\mathbf{x}_j^{(\ell)} = \sigma \left(\sum_{i=1}^{F_{\ell-1}} \mathbf{Y}_{ij}^{(\ell)} \mathbf{x}_i^{(\ell-1)} + \mathbf{b}_j^{(\ell)} \right) \tag{20}$$

where F_ℓ is the number of channels in layer ℓ, and σ is a complex version of ReLU.

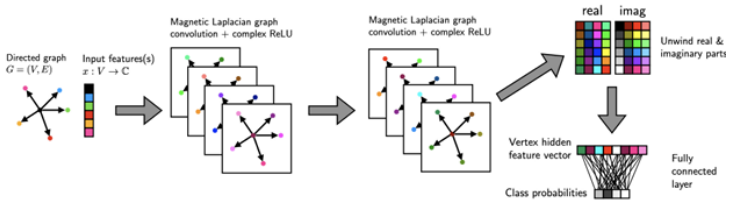

Figure 1: MagNet (L=2) applied to node classification

5 Node Classification Problem

Given a graph $G = (V, E)$ with an adjacency matrix $A \in \{0, 1\}^{|V| \times |V|}$ and node features $X \in \mathbb{R}^{|V| \times d}$, where each node $v \in V$ has a feature vector x_v, the objective is to classify each node into one of C classes.

We define a function $f : X, A \to Y$ that predicts node labels $\hat{y}_v$ for $v \in V$ by minimizing the classification loss:

$$\mathcal{L} = \sum_{v \in V_L} \ell(f(X, A)_v, y_v) \tag{21}$$

where $V_L \subset V$ is the labeled set and ℓ is the cross-entropy loss:

$$\ell(y_v, \hat{y}_v) = -\sum_{c=1}^{C} \mathbf{1}[y_v = c] \log P(y_v = c | X, A) \tag{22}$$

In our case, we have pre-processed the spatial traffic data and have tested 3 different types of classes: a 2-bit binary class - one for traffic into the node, and one for traffic from it, a 1-bit binary class for traffic into the node, and another 1-bit binary class for traffic from it.

6 Link Predication Problem

The goal of link prediction is to estimate the probability of an edge $(u, v) \in E$.

Given node embeddings $Z \in \mathbb{R}^{|V| \times d}$, we define a function $g : V \times V \to [0, 1]$:

$$\hat{A}_{uv} = g(z_u, z_v) \tag{23}$$

where $\hat{A}_{uv}$ is the predicted probability of an edge.

The objective is to minimize the binary cross-entropy loss:

$$\mathcal{L} = -\sum_{(u,v) \in E} \log g(z_u, z_v) - \sum_{(u,v) \notin E} \log(1 - g(z_u, z_v)) \tag{24}$$

The edges between nodes can change based on some parameter, in our case, over time - estimate the likelihood of traffic flow between segments over time.

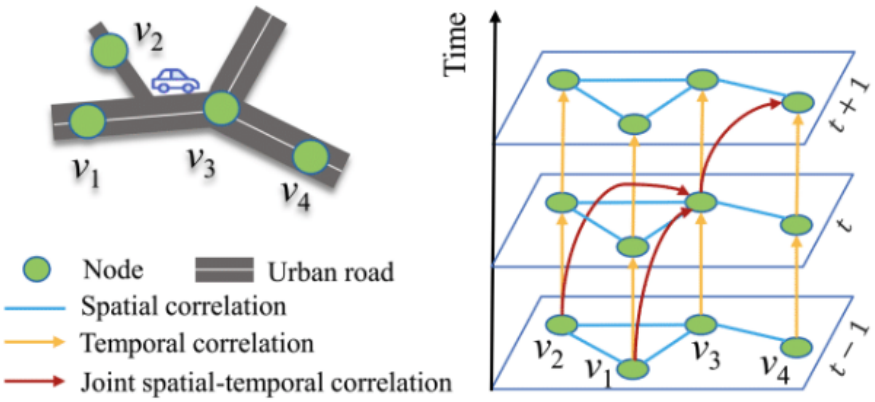

Figure 2: Enter Caption

Since our data is a weighted spatial-temporal data, we used this problem to try and predict the weight of each edge, meaning predicting the average speed on each road based on the time and day.

7 Temporal Graph Construction

For the link prediction task, we had to re-build our graphs. This time, we split the data into 18 different 3-hour time slots across the 3 days. For each time slot, we created a directed graph in the following way: Nodes are unique (longitude, latitude) coordinates, edges represent taxi trips between locations while edge attributes includes: Weight: Speed of the trip (calculated as distance/time), Date and time of the pickup. Than, we had to find common nodes across all 18 time slot graphs (total of 129 common nodes), removing all nodes not present in all graphs. This way, we had the same nodes on each time slot, with different edges - rides, and weights - speed. Then, we connected all graphs into one big graph, adding ID to each node to get nodes with the same location remain different from each other (representing the connection shown as yellow edges on figure 1, even though there is no actual edge between them). In this way we got a spatial-temporal directed graph representing our data. As a starting point we didn't connect nodes between two different time slots, to make the structure of the graph simpler (only blue edges on figure 1), but adding a cross-time slot rides is an interesting option for future work - a ride started at one slot and ended at the other, the edge will be cross-time slots (red edges in figure 1).

8 Datasets

First, we searched for benchmarks or other known datasets consisting of temporal directed traffic data. We used "Graph Neural Network for Traffic Forecasting: A Survey" (Reference 1) which covers the main traffic forecasting problems in research, a survey of papers suggesting state-of-the-art solutions that solve these problems, and a variety of datasets used in the different papers. Two main datasets we first tried were PeMSD8 [5] and metr-la [6], both based on traffic sensors in the USA. All pre-processed dataset based on them, were actually not directed - adjacency matrices were symmetric, and other data tables were lack of direction [5]. Hence, we tried a different type of traffic dataset - Taxis. Finally, we used the NYC taxi dataset [4] collected in January 2016. It have the following trip elements: Trip starting and ending point (longitude + latitude), trip duration, date, and time. From the data, we extracted the graph nodes and edges, where nodes are locations, edges are trips, and the weights are the average speed of the trip. Locations were calculated as a round of 2.5 of the long/lat values for nodes, and distances were calculated as air distance in meters. This way, we gathered points to the closest junction as one node in the graph. This way, we eventually got both the graph and features matrix as input to the network.

9 Results

We evaluated the performance of our approach using both symmetric and non-symmetric adaptations of the MagNet [1] model. Since MagNet [1] has not been previously applied to weighted directed graphs, we incorporated this information into the magnetic adjacency matrix using these two methods. The key difference is:

- Symmetric version: Directional information is included only in the phase matrix. The weighted adjacency matrix is used to compute the phase matrix, while the symmetric adjacency matrix is derived from the non-weighted adjacency matrix. - Non-symmetric version: Both the phase matrix and the symmetric adjacency matrix are computed using the weighted adjacency matrix, preserving more directional information.

We tested both methods on traffic flow forecasting tasks using 1-bit label and 2-bit label classification settings. The classification accuracy for both training and test sets is summarized in Table 1.

The results indicate that the symmetric version of MagNet [1] consistently achieved higher accuracy in the 1-bit label classification tasks, with test accuracies of 86.0% (into) and 85.9% (from), compared to the non-symmetric version, which achieved 84.8% and 84.1%, respectively. For the 2-bit label classification, performance was slightly lower, with test accuracy around 80% in both approaches.

To better illustrate the differences, Figure 3 shows the classification performance for the non-symmetric version, symmetric version, and a combined

Method	Labels	Train Accuracy	Test Accuracy
Non-Symmetric	into	85.7%	84.8%
	from	-	84.1%
	into + from	84.7%	80.0%
Symmetric	2 into	86.9%	86.0%
	2 from	84.5%	85.9%
	into + from	82.8%	80.3%

Table 1: Comparison of train and test accuracy for symmetric and non-symmetric adaptations of MagNet.

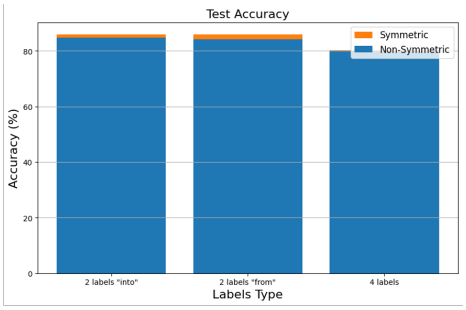

Figure 3: esults for Class Prediction Problem

comparison of both approaches, respectively.

Overall, these findings suggest that incorporating a symmetric adjacency matrix improves the classification performance of MagNet [1] for traffic flow forecasting, particularly in the 2-label setting. Future work could explore alternative ways to represent directed and weighted information within the magnetic Laplacian framework to further enhance predictive performance.

10 Conclusions

In this work, we explored various graph-based approaches and deep learning techniques for traffic flow forecasting, focusing on node classification and link prediction within spatial-temporal data. Our analysis covered multiple real-world datasets, including PeMSD8 [5], METR-LA [6], and NYC Taxi [4], where we investigated different data processing and representation methods to effectively capture traffic dynamics.

We evaluated three key models—Magnetic Laplacian GNN [1], DyRep [2], and ST-SSL [3]—analyzing their strengths and limitations in this context. Our findings suggest that graph-based methods provide a strong foundation for traffic modeling, leveraging the inherent structure of transportation networks.

Overall, our study highlights the potential of these approaches for traffic forecasting, emphasizing the importance of appropriate data representation and

model tuning. Future work could further refine these techniques by incorporating additional spatiotemporal dependencies and exploring more advanced learning paradigms.

References

[1] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, and M. Hirn. Magnet: A neural network for directed graphs. *Advances in Neural Information Processing Systems*, 34:27003–27015, 2021.

[2] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha. Dyrep: Learning representations over dynamic graphs. *arXiv preprint*, 2021.

[3] Anonymous. Spatio-temporal self-supervised learning for traffic forecasting. *OpenReview*, 2021.

[4] P. Rohan. Nyc taxi trip duration dataset, 2021.

[5] AmitRoy7781. Sst-gnn pemsd8 dataset, 2021.

[6] A. Nguyen. Metr-la dataset, 2021.