Directed GNNs for Traffic Forecasting

Annette Jing
ajing@stanford.edu

Myra Deng
myradeng@stanford.edu

This report is built off of the proposal. To facilitate easier grading, we highlight sections about our current progress with sections in blue headings that start with "Milestone implementations."

Our project repository is: `https://github.com/AnnetteJing/CS224W_Project/tree/main`

To preface, we chose to utilize an existing package for time series forecasting with GNN called Pytorch Geometric Temporal (`pyg-temporal`) [1], because it provides a modeling framework that unifies many methods in this domain.

1 Dataset

1.1 Application domain

This project focuses on the application of Graph Neural Networks (GNNs) to the domain of traffic prediction, with special attention given to different ways of encoding directionality in the graph convolution. Traffic prediction is a critical component of intelligent transportation systems, enabling better traffic management, route planning, and urban infrastructure development. By accurately forecasting traffic conditions, we can contribute to reducing congestion, improving travel time reliability, and enhancing overall urban mobility. We focus on the role of directionality because it naturally arises in this task – up and downstream traffic affect future traffic conditions differently.

1.2 Dataset

We plan to use two benchmark datasets for our task. We chose these datasets as they capture rich, real-world temporal data and are commonly used in the literature for traffic prediction models [2] [3].

- **METR-LA** (Los Angeles Metropolitan Traffic)
  This dataset contains traffic network information for Los Angeles city, collected from March to June 2012 at 5 minute intervals for a total of 34,272 samples with a missing ratio of 8.109%. The data is collected from 207 sensors on the LA County road network.
- **PEMS-BAY** (Bay Area Performance Measurement System)
  This dataset is collected by CalTrans Performance Measurement System and is recorded at 5 minute intervals from January to March 2017 for a total of 52,116 samples with a missing ratio of 0.003%. The data is collected from 325 sensors on the Santa Clara County road network.

1.3 Task and Metrics

Traffic flow prediction can be framed as a spatio-temporal forecasting challenge for regularly sampled multivariate time series (MTS) data. In terms of GNN training, this is a supervised node-level regression problem. The dataset consists of observations $\mathbf{X}_{1:T} \equiv \{\mathbf{X}_1, ..., \mathbf{X}_T\} \in \mathbb{R}^{N\times F\times T}$ over $t = 1, ..., T$ timesteps, where the signal $\mathbf{X}_t := [\mathbf{x}_{1,t}, ..., \mathbf{x}_{N,t}]^\intercal \in \mathbb{R}^{N\times F}$ at each timestep t contains F observed features (e.g. speed, velocity, volume, etc) for each of the N time series. Further, the time series are related by a potentially time-varying and/or unknown graph $\mathcal{G} := \{\mathcal{G}_1, ..., \mathcal{G}_T\}$, $\mathcal{G}_t := (\mathcal{V}, \mathcal{E}_t)$ with $|\mathcal{V}| = N$ nodes corresponding to the N series.

Given a target feature mapping $y : \mathbb{R}^F \to \mathbb{R}^C$, look-back window W, and forecast horizon H, the MTS forecasting task is to learn a function $f : \mathbb{R}^{N\times F\times W} \to \mathbb{R}^{N\times C\times H}$ that maps the historical features to future targets:

$$\mathbf{X}_{t-W+1:t} \equiv [\mathbf{X}_{t-W+1} \quad \cdots \quad \mathbf{X}_t] \overset{f}{\longmapsto} [\mathbf{Y}_{t+1} \quad \cdots \quad \mathbf{Y}_{t+H}] \equiv \mathbf{Y}_{t+1:t+H},$$

where we abuse notations and write $\mathbf{Y}_{t+h} := y(\mathbf{X}_{t+h}) \equiv [y(\mathbf{x}_{1,t+h}), ..., y(\mathbf{x}_{N,t+h})]^\intercal \in \mathbb{R}^{N \times C}$. If an input graph $\mathcal{G}$ is given, f can take it into account. Otherwise, it would have to learn an implied graph structure based on the signals $\mathbf{X}_{t-W+1:t}$ alone. In our case, both METR-LA and PEMS-BAY come with a time-static graph defined by the location of traffic sensors.

Given our data is collected at 5-minute intervals, 12 steps ahead is equivalent to $5 \cdot 12 = 60$ minutes into the future. For evaluation, we use the standard metrics employed in traffic prediction GNN studies, specifically the *mean absolute error* (MAE), *root mean square error* (RMSE), and *mean absolute percentage error* (MAPE) up to 12 steps ahead, that is, $H = 12$. For easy comparison with existing literature (see Section 2.2), we set the look-back window to $W = 12$ but will consider expanding the window if time allows. Missing values are excluded in calculating these metrics.

1.4 Milestone implementations: Dataset

`https://github.com/AnnetteJing/CS224W_Project/blob/main/src/utils/data_utils.py`

For the dataset, we built a wrapper around the `pyg-temporal` dataloader objects and made minor modifications to the dataloaders for our selected datasets (`METRLADatasetLoader` and `PemsBayDatasetLoader`). There are two reasons for this:

1. The batching functionalities of `pyg-temporal` dataloaders are limited. We preprocess the dataset in the `TimeSeriesDataset` wrapper class for simpler batching.
2. The normalization performed by `pyg-temporal` dataloaders use the entire dataset instead of just the training dataset, raising concerns about data leakage. To fix this problem, we removed the normalization step from the `pyg-temporal` dataloaders, and only create a "`Scaler`" object after train-test split using the training dataset. The `Scaler` object can normalize and unnormalize any given batch using the training dataset's mean and standard deviation. This matches what is done in the paper repository of DCRNN [4].

1.5 Milestone implementations: Evaluation Metrics

`https://github.com/AnnetteJing/CS224W_Project/blob/main/src/utils/metrics.py`

We implemented an `Evaluator` class that keeps track of the sum of square error (SSE), sum of absolute error (SAE), sum of absolute percentage error (SAPE), as well as the number of samples used (dropping all missing values, and zeros in the case of SAPE) for every node (series) and horizon (steps into the future). This allows us to quickly compute the RMSE, MAE, and MAPE at any point during training or evaluation.

Further, since the error for each series is recorded individually, we can look into aggregations other than the average across series. For example, the higher quantiles of the errors can provide insight into the worst case performance of a model.

2 Methodology

2.1 Model Details

Models for GNN time series forecasting can mostly be separated into a *graph module* that captures inter-series relationships and a *temporal module* that propagates information across time. We plan to investigate the application of *directed spectral graph convolutions* to the task of traffic forecasting, which pertains to the graph module.

2.1.1 Graph Module

Generally, spectral graph convolutions are only defined for undirected graphs due to its underlying theory being rooted in viewing the graph as a non-Euclidean space with graph-defined distances, and distance metrics are symmetrical by definition. More precisely, a traditional spectral graph convolution on undirected graphs is defined as follows:

1. Construct a symmetric *graph Laplacian* $\mathbf{L}$ analogous to the Laplace operator Δ (defined by $\Delta f := \sum_{n=1}^{N} \frac{\partial^2 f}{\partial x_n^2}$) in that both measure the divergence of a signal from its neighboring average.
2. Orthogonally diagonalize $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\dagger$ to obtain a *graph Fourier basis*, $\mathbf{U}$, analogous to the Fourier basis because the latter are eigenfunctions of the Laplace operator – $\Delta e^{2\pi i s t} = -4\pi^2 |s|^2 e^{2\pi i s t}$.

3. The *graph Fourier transform* & *inverse transform* are defined as $\mathcal{F}_{\mathcal{G}} f := \mathbf{U}^\dagger f$ and $\mathcal{F}_{\mathcal{G}}^{-1} f := \mathbf{U} f$ since the Fourier transform is the inner product between input signals and the Fourier basis.
4. By the convolution theorem ($f \star g = \mathcal{F}^{-1}(\mathcal{F} f \cdot \mathcal{F} g)$), the *spectral graph convolution* of a signal $\mathbf{x} \in \mathbb{R}^N$ and a filter $\theta \in \mathbb{R}^N$ is defined by

$$\mathbf{x} \star_{\mathcal{G}} \theta := \mathbf{U}\big(\mathbf{U}^\dagger \mathbf{x} \odot \mathbf{U}^\dagger \theta\big) = \mathbf{U}\mathrm{diag}\big(\mathbf{U}^\dagger \theta\big)\mathbf{U}^\dagger \mathbf{x}.$$

   To improve efficiency, this operation is often approximated by replacing $\mathrm{diag}(\mathbf{U}^\dagger \theta)$ with a polynomial of Λ, $\mathrm{poly}_\theta(\Lambda)$, in which case we have $\mathbf{x} \star_{\mathcal{G}} \theta = \mathrm{poly}_\theta(\mathbf{L})\mathbf{x}$ [5] [6].

For a directed graph, the 2nd step is generally infeasible because a real square matrix is orthogonally diagonalizable if and only if it is symmetric. To solve this, we consider the following:

SVD & self-adjoint dilation. Adhering to convention, we use the normalized directed graph Laplacian $\mathbf{L} := \mathbf{I}_N - \mathbf{D}_{\text{in}}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}_{\text{in}}^{-\frac{1}{2}}$. While $\mathbf{L}$ is asymmetric, it always has a singular value decomposition (SVD), and directionality can be captured by the left and right singular vectors. In fact, the SVD is equivalent to performing eigen-decomposition on the dilation of $\mathbf{L}$ [7]:

$$\mathcal{S}(\mathbf{L}) := \begin{bmatrix} \mathbf{0} & \mathbf{L} \\ \mathbf{L}^\intercal & \mathbf{0} \end{bmatrix} = \mathbf{I}_{2N} - \begin{bmatrix} \mathbf{D}_{\text{in}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\text{in}} \end{bmatrix}^{-\frac{1}{2}} \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^\intercal & \mathbf{0} \end{bmatrix}}_{=:\mathcal{S}(\mathbf{A})} \begin{bmatrix} \mathbf{D}_{\text{in}} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\text{in}} \end{bmatrix}^{-\frac{1}{2}}$$

Note that if we replace the first $\mathbf{D}_{\text{in}}$ by $\mathbf{D}_{\text{out}}$ then it is equivalent to the Laplacian of $\mathcal{S}(\mathbf{A})$.

Moralized dilation. Inspired by contemporaneous interactions in vector autoregressive type models as well as "directed graph moralization" in probabilistic graphical model theory [8], we consider using the eigen-decomposition of the Laplacian of a "moralized dilation" of $\mathbf{A}$,

$$\mathcal{M}_c(\mathbf{A}) := \begin{bmatrix} c\mathbf{A}\mathbf{A}^\intercal & \mathbf{A} \\ \mathbf{A}^\intercal & \mathbf{0} \end{bmatrix} \quad \text{where } c \text{ is a scaling constant such as } \frac{1}{\sigma_{\max}(\mathbf{A})}.$$

This will result in a different weighing of the left and right singular vectors compared to SVD or the eigen-decomposition of the Laplacian of $\mathcal{S}(\mathbf{A})$, allowing for more mixing in upstream affects.

Magnetic signed Laplacian. If time allows, we may look into this alternative directed spectral convolution defined in the complex space [9] that has promising results in graph learning tasks like node clustering, but has not yet been applied to multivariate time series forecasting.

2.1.2 Temporal Module

Our candidates for the temporal module include the Gated Recurrent Unit (GRU) [10] [11] [12]

$$\begin{aligned}
\mathbf{Z}_t &= \sigma\big([\mathbf{X}_{t-W+1:t}, \mathbf{H}_{t-1}] \star_{\mathcal{G}} \Theta_{\mathbf{Z}} + \mathbf{B}_{\mathbf{Z}}\big), \\
\mathbf{R}_t &= \sigma\big([\mathbf{X}_{t-W+1:t}, \mathbf{H}_{t-1}] \star_{\mathcal{G}} \Theta_{\mathbf{R}} + \mathbf{B}_{\mathbf{R}}\big), \\
\mathbf{C}_t &= \tanh\big([\mathbf{X}_{t-W+1:t}, \mathbf{R}_t \odot \mathbf{H}_{t-1}] \star_{\mathcal{G}} \Theta_{\mathbf{C}} + \mathbf{B}_{\mathbf{C}}\big), \\
\mathbf{H}_t &= \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \mathbf{C}_t,
\end{aligned}$$

and attention-based methods [13] [14] [15]. Unless suggested otherwise, we plan to start with GRU as its structure is more standardized.

2.1.3 Other modeling & training details

Training-wise, we will be using time-series cross validation, aka backtesting. This is different from transductive data splitting because we are splitting in the time dimension rather than the graph, which means all nodes features and labels are observed in each split. Similar to transductive data splitting, however, While the datasets include input graphs, we may explore implicit graph modeling through node embeddings [16] or other parameterizations [12], and also consider blending the given graph and implicit graph. The input graph can be seen at every time step, but an implicit graph may be evolving over time.

2.2 Baselines & Other Competing Methods

For comparison, we shall report the performance of traditional statistical methods (e.g. historical average and ARIMA), undirected spectral GNNs (e.g. STGCN [6]), directed spatial GNNs (e.g. DCRNN [4], which uses diffusion convolution to capture directionality and is equivalent to the undirected spectral graph convolution defined in Section 2.1.1 when the input graph is undirected).

3 Milestone implementations: Preliminary Results

3.1 Baseline DCRNN

We re-implement DCRNN [4] using the pytorch geometric temporal library to introduce a comparable baseline for our spectral convolution GNN implementation. We replicated most of the same hyperparameters as the original paper and report the same metrics. After significant debugging, we were able to get reasonable results, but haven't had time to tune the hyperparameters further to better match the paper's reported performance.

3.2 SVD Spectral Conv GNN

We also implemented a version of the spectral convolution GNN with SVD & self-adjoint dilation (see Github repo for model code and Section 2.1.1 for implementation details) to compare against this baseline, and are in the process of training this model.

3.3 Preliminary Results

	T	Metric	DCRNN	DCRNN (re-impl.)	Spectral-RNN
METR-LA	15 min	MAE	2.77	-	-
		RMSE	5.38	-	-
		MAPE	-	-	-
	30 min	MAE	3.15	-	-
		RMSE	6.45	-	-
		MAPE	8.8%	-	-
	1 hour	MAE	3.60	-	-
		RMSE	7.59	-	-
		MAPE	10.5%	-	-
PEMS-BAY	15 min	MAE	1.38	5.33	-
		RMSE	2.95	8.89	-
		MAPE	2.9%	14.43%	-
	30 min	MAE	1.74	6.00	-
		RMSE	3.97	8.89	-
		MAPE	3.9%	15.11%	-
	1 hour	MAE	2.07	5.60	-
		RMSE	4.74	8.88	-
		MAPE	4.9%	14.64%	-

Table 1: Performance comparison between model variants

References

[1] B. Rozemberczki, P. Scherer, Y. He, G. Panagopoulos, A. Riedel, M. Astefanoaei, O. Kiss, F. Beres, G. Lopez, N. Collignon, and R. Sarkar, "PyTorch Geometric Temporal: Spatiotemporal signal processing with neural machine learning models," *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, p. 4564–4573, 2021.

[2] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, vol. 26, pp. 753–763, 2020.

[3] M. Jin, H. Y. Koh, Q. Wen, D. Zambon, C. Alippi, G. I. Webb, I. King, and S. Pan, "A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1–20, 2024.

[4] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[5] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Proceedings of the 30th Conference on Neural Information Processing Systems*, 2016.

[6] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018.

[7] Y. Chen, C. Cheng, and Q. Sun, "Graph Fourier transform based on singular value decomposition of the directed Laplacian," *Sampl. Theory Signal Process. Data Anal.*, vol. 21, no. 24, 2023.

[8] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principle and Techniques*. Cambridge, MA: MIT Press, 2009.

[9] Y. He, M. Perlmutter, G. Reinert, and M. Cucuringu, "MSGNN: A spectral graph neural network based on a novel magnetic signed laplacian," *Proceedings of the 1st Learning on Graphs Conference*, 2022.

[10] H. Yu, T. Li, W. Yu, J. Li, Y. Huang, L. Wang, and A. Liu, "Regularized graph structure learning with semantic knowledge for multi-variates time-series forecasting," *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, 2022.

[11] Y. Chen, I. Segovia-Dominguez, and Y. Gel, "Z-GCNETs: Time zigzags at graph convolutional networks for time series forecasting," *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[12] C. Shang, J. Chen, and J. Bi, "Discrete graph structure learning for forecasting multiple time series," *Proceedings of the 9th International Conference on Learning Representations*, 2021.

[13] G. Wang, Y. Chen, M. Gao, Z. Wu, J. Tang, and J. Zhao, "A time series is worth five experts: Heterogeneous mixture of experts for traffic flow prediction," *arXiv 2409.17440*, 2024.

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

[15] Y. Xia, Y. Liang, H. Wen, X. Liu, K. Wang, Z. Zhou, and R. Zimmermann, "Deciphering spatio-temporal graph forecasting: A causal lens and treatment," *Proceedings of the 37th Conference on Neural Information Processing Systems*, 2023.

[16] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020.