Learning From Networks: Final Report

Using Graph Neural Networks and Deep Learning
for traffic forecasting and predicting speed

Cristian Boldrin, Alberto Makosa, Simone Mosco

January 2022

Abstract

The significant increase in world population and urbanisation has brought several important challenges, in particular regarding the sustainability, maintenance and planning of urban mobility. In this paper, we will try to deal with the task of traffic forecasting; in particular we aim to use historical speed data to predict the speed at a future time step. Other generic variables we can focus on in order to make predictions are: volume, density, flow, congestion condition, and occupancy. Due to high non-linearity, randomness and complexity of the traffic flow, we will try to use Graph Neural Networks to deal with long-term and mid-term predictions.

1 Introduction

Transportation plays a vital role in everybody's routine. According to a survey in 2021 [11], 8 % of U.S. residents ages 16 years older drove at least occasionally in 2021. Drivers reported making an average of 2.57 driving trips, spending 61.3 minutes behind the wheel, and driving 32.7 miles each day in 2021, all of which represented substantial increases relative to past years. Moreover, it is worth to care the explosion of traffic in the U.S. in the recent years [8]. Under this circumstance, accurate real-time forecast of traffic conditions is of paramount importance for road users, private sectors and governments. Where traffic forecasting is not only based on consideration of historical traffic state data, but also on external factors that may affect traffic state, a simple example is the time of the day at which the road is travelled, or even the time of the year, on holidays the traffic will probably get worse. As a matter of fact, the prediction made by the traffic forecasting can be classified with respect to the length of the prediction, that is we consider short-term (5-30 min) and medium and long term (over 30 min). From the literature already present it is easy to see that for short-term prediction statistical approaches (e.g., linear regression) are able to get good results with good computational time. However, as the time increase and with it the complexity, but most importantly the uncertainty of traffic flow. these methods are not able to get acceptable prediction.
In order to try and solve the medium and long term traffic forecasting we tried to use several techniques based on Convolutional Neural Network and Graph Neural Network. In particular we focused on Temporal Convolutional Layer, a framework which employs casual convolutions and dilations so that it is adaptive for sequential data with its temporality and large receptive fields, on Spatial Convolutional Layer, which uses spatial features of nodes to aggregate information from the neighbors, and on Spatio-Temporal Convolution Block, which is a combination of the previous twos.

1.1 Related works

In recent years, to model the graph structures in transportation systems, graph neural networks have been introduced and have achieved state-of-the-art performance in a series of traffic forecasting problems. As a consequence, this had become an inspiration for several great works done by many researchers, which had achieved excellent result for specific traffic forecasting problems. In our paper we couldn't use all of them to improve our proposal; however, we still would like to cite some of them for their groundbreaking intuition, like **Graph Neural Networks for Traffic Forecasting** [7], which is an overview of different approaches for traffic forecasting, from traditional ones to deep learning models, including Recurrent or Convolutional Graph Neural Network, Graph Attention Network, Graph Autoencoders and STGCN, or like in **Structured Sequence Modeling with Graph Convolutional Recurrent Networks** [3], which introduces Graph Convolutional Recurrent Network (GCRN), a deep learning model able to predict structured sequences of data, where they describe a GCRN as a generalization of classical recurrent neural networks (RNN) to data structured by an arbitrary graph.

2 Problem Definition

Traffic forecast is a typical time-series prediction problem, that consists in predicting the most likely traffic measurements (e.g. speed or traffic flow) in the next H time steps given the previous M observations from traffic network G, as:

$$\hat{v_{t+1}}, ..., \hat{v_{t+H}} = \text{argmax}_{v_{t+1},...,v_{t+H}} \log P(v_{t+1}, ..., v_{t+H} \mid v_{t-M+1}, ..., v_t) \quad (1)$$

where $v_t \in \mathbb{R}^n$ is an observation vector of n roads segments covered by n sensors at time step t, each element of which records historical observation for a single road segment. We define the traffic network on a graph: the observation v_t is not independent but linked by pairwise connection with weights, as show in Fig. 1. So, at time t, the graph can be expressed as $G_t = (V_t, E, W)$ with:

- $V_t \in \mathbb{R}^n$ set of vertices which labels correspond to observation from n sensors at time t (e.g., a speed measurement)
- $E \in \mathbb{R}^n$ x $\mathbb{R}^n$ set of edges, indicating the connections between stations
- $W \in \mathbb{R}^{nxn}$ is the weighted adjacency matrix, indicating the distances between sensors

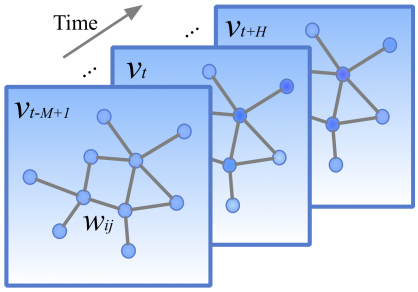


Figure 1: Graph-structured traffic data. Each v_t indicates a frame of current traffic status at time step t, which is recorded in a graph-structured data matrix.

Taking inspiration by [14], the approach we used to generalize CNNs and adapt them to structured data graphs is to expand the spatial definition of a convolution [9] and adding temporal layers capable of extracting features. Several works [10], [12] studied how to make spatial graph convolution more promising by reducing the computational complexity from $O(n^2)$ to nearly linear, using basically first order approximations of Chebyshev polynomials.

3 Proposed Model

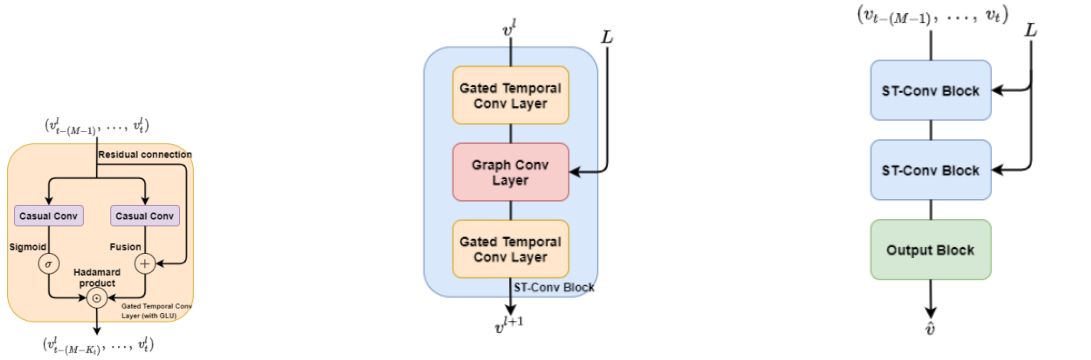
3.1 Temporal Convolution Layer

Although RNN-based models become widespread in time-series analysis, recurrent network for traffic prediction still suffer from time-consuming iterations, complex gate mechanism, and slow response to dynamic changes. Moreover, even using pretty small time windows both in the past and the future, recurrent networks tend to have problems when dealing with past-term dependencies due to gradient vanishing, given that our weight matrix is contractive, i.e., its spectral radius (that corresponds to its largest eigenvalue) is smaller than 1 [4]. In the literature, common approaches to try to cope or reduce somehow vanishing gradients problem is using extensions of the simple RNN such as LSTM, Gated Recurrent Units or Reservoir Computing idea. On the other hand, CNNs have the advantages of faster training, simpler structures, and no dependency constraints to previous steps.

Our original idea was to implement LSTM or GRU architectures for extracting meaningful temporal features, but we get lost with implementation issues and details, so we decided to stick to the original idea of our main reference paper with some minor modifications. As shown in figure 2a, the temporal convolutional layer is composed by an Align layer, used to align the input channel numbers and the output channel numbers, and a Causal Convolutional two-dimensional layer that takes the convolution with the input signal and the temporal kernel, doubling the output dimension without padding. In particular, Causal convolution is a variant of standard convolution in which the output of a neuron at a given time step is only dependent on the input at that time step and any previous time steps, and not depending on future time steps. This is beneficial when processing sequential data, since it helps to prevent the model from "cheating" by using information from the future when making predictions. Then, a gated linear unit (GLU) is introduced as a non-linearity, that corresponds to:

$$\text{GLU}(x) = P \odot \sigma(Q) \quad (2)$$

where x is the input signal, P and Q are output of the 2D convolution splitted in half with the same size of channels, $\odot$ denotes the element-wise Hadamard product. The sigmoid gate $\sigma(Q)$ controls which input P of the current state are relevant for discovering compositional structure and dynamic variances in time series and contributes to exploiting the full input through stacked temporal convolutional layers.



(a) Temporal Layer (b) Architecture of a ST block (c) Architecture of STGCN

Figure 2: Architecture of spatio-temporal graph convolutional networks

3.2 Spatial Convolution Layer

Since a standard convolution for regular grids is not applicable to general graphs, there are two basic approaches in the literature exploring how to generalize CNNs to structured data-types:

- expand the definition of a convolution [9], that rearranges the vertices into certain grid forms which can be processed by a standard convolutional operator
- manipulate the spectral domain with graph Fourier transform [6], dealing with spectral framework to apply convolution

Based on these two approaches, we introduce the notion of graph convolution operator $*_{\mathcal{G}}$ as the multiplication of a signal $x \in \mathbb{R}^n$ with a kernel Θ:

$$\Theta *_{\mathcal{G}} x = \Theta(L)x = \Theta(U\Lambda U^T)x = U\Theta(\Lambda)U^T x \tag{3}$$

where graph Fourier basis $U \in \mathbb{R}^{nxn}$ is the matrix of eigenvectors of the normalized graph Laplacian $L = I_n - D^{-\frac{1}{2}}WD^{-\frac{1}{2}} = U\Lambda U^T \in \mathbb{R}^{nxn}$ ($D \in \mathbb{R}^{nxn}$ is the diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$), $\Lambda \in \mathbb{R}^{nxn}$ is the diagonal matrix of eigenvalues of L_i and filter $\Theta(\Lambda)$ is also a diagonal matrix. By this definition, a graph signal x is filtered by a kernel Θ with multiplication between Θ and graph Fourier transform $U^T x$. [1]

The traffic network is organized as a graph structure, where sensors measuring speed represents the nodes and the roads (i.e., links between sensors) are the edges. However, previous studies neglect spatial attributes of traffic networks and they split the graph into several segments or grids, that can capture only rough spatial local features. Accordingly, following [14], we employ the graph convolution directly on the whole graph in order to extract meaningful patterns and features in the space domain. Since the computation of Θ in Eq. 3 can be expensive due to $\mathcal{O}(n^2)$ multiplications with graph Fourier basis, we introduce the Chebyshev Polynomials Approximation to reduce complexity.

To localize the filter and reduce the parameters, the kernel Θ can be restricted to a polynomial of Λ as $\Theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$, where $\theta \in \mathbb{R}^K$ is a vector of polynomials coefficients, K is the kernel size, which determines the maximum radius of the convolution from central nodes. Chebyshev polynomial $T_k(x)$ is used to approximate kernels as a truncated expansions of order $K-1$ as $\Theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda})$, with re-scaled $\tilde{\Lambda} = 2\Lambda/\Lambda_{max} - I_n$ (Λ_{max} is the largest eigenvalue of L) [5]. Hence, graph convolution can be rewritten as:

$$\Theta *_{\mathcal{G}} x = \Theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x \tag{4}$$

where $T_k(\tilde{L}) \in \mathbb{R}^{nxn}$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\tilde{L} = 2L/\lambda_{max} - I_n$. As show by [10], the cost of Eq. 3 can be reduced to $\mathcal{O}(K|\mathcal{E}|)$ with Eq. 4.

In our implementation of the spatial convolutional layer, we generalized the graph convolution and we implemented it using Einstein summation following this code.

3.3 Spatio-temporal Convolutional Block

As we can see in Fig. 2b, the spatio-temporal convolutional block (ST-Conv-block) is constructed including a spatio-convolutional layer in between two temporal-convolutional layer, in order to merge features from both spatial and temporal domains. This "bottleneck" approach also helps the network to achieve scale compression and feature squeezing by downscaling and upscaling of channels C through the graph convolutional layer. Furthermore, layer normalization is used at the end of every block in order to prevent overfitting and parameter explosing. After stacking b ST-Conv-blocks, we add an Output Block comprising a Temporal Convolutional Layer, a Normalization Layer and two Fully-Connected Layer that are used to make the final single-time step prediction, as in Fig. 2c.

Thus, the model is entirely composed of convolutional structures and therefore achieves parallelization and faster training speed. More importantly, this economic architecture allows the model to handle large-scale networks with more efficiency.

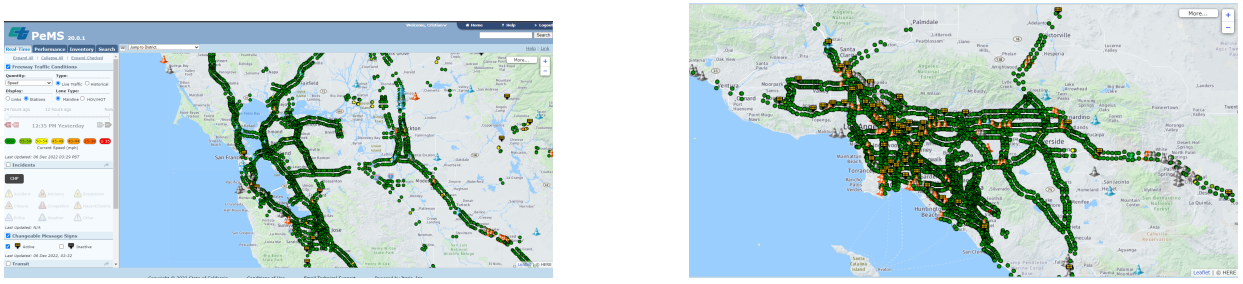
4 Experiments

4.1 Dataset description

We will focus mainly on the following datasets, as shown in 3:

- PeMS dataset, containing raw measurements from over 39,000 sensors stations on the freeway system spanning all major metropolitan areas of California, from 2001 up until now, available on here. We considered district number 7 of LA area and number 4 of Bay Area (Fig. 3a) being the most representative metropolitan areas for experiments of this type. For downloading the data, we select 5-minutes slot time measurements (i.e., 12 slots per hour, 288 slots per day) and then we can select the total time window of historical data.
- MetrLA dataset (Fig. 3b) available here, containing a total of 119 days aggregating in 5-minutes slot measurement (as before) for 208 sensors in the metropolitan area of Los Angeles, so resulting in a [34.272, 207]-shaped array.

In both cases, we will consider only workdays for our experiments [13].



(a) Bay area (district 4) map

(b) MetrLA dataset (district 7) map

Figure 3: Dataset sensors and roads map

4.2 Data Preprocessing

Firstly, we create the adjacency matrix from the dataset in a slightly different way with respect to the main reference paper we followed. In particular, we followed [10] implementation from the ChebNet as:

$$w_{ij} = \begin{cases} e^{\frac{-d_{ij}^2}{\sigma^2}} & \text{if } d_{ij} \leq k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where w_{ij} represents the edge weight from vertex i to vertex j, d_{ij} is the Euclidean distance from vertex i to vertex j, σ is the standard deviation of the distances and k is the threshold for inducing sparsity in the weighted adjacency matrix. Then, we scaled the data using a standard normalization scaler, and we prepare the data in windows based on the historical data to use and the number of steps in the future to predict.

4.3 Experiment Result

For training, we use MSE loss function, defined as:

$$\mathcal{L}(\hat{v}; W_\theta) = \sum_t ||\hat{v}(v_{t-M+1}, ..., v_t, W_\theta) - v_{t+H}||^2 \tag{6}$$

where W_θ are all trainable parameters in the model, v_{t+H} is the ground truth and $\hat{v}(\cdot)$ denotes the model's prediction.

We trained our model on a laptop with NVIDIA GeForce RTX 3060 (CUDA compatibility = 8.6) in Windows 11 inside Conda environment. An epoch of training, thanks to Eq. 4 approximation, takes $\sim$ 70 sec. We executed grid search strategies to find the best parameters on validation. In particular, we find out the best settings are achieved with 2 ST-Conv-blocks, Temporal kernel size = Spatial kernel size = 2, batch size = 32, learning rate = 0.005, dropout probability = 0.5. Furthermore, we use AdamW optimizer in combination with learning rate scheduler to scale the network parameters. Finally, we introduce Early Stopping with patience on number of epochs = 20, to stop the model if no improvements are noted.

As regarding the performance evaluation, we will focus on Mean Absolute Error, Mean Absolute Percentage Error and Root Mean Squared error as principal metrics of evaluation. We will compare our performance with naive predictors and current state of the art for the datasets [2].

4.4 Naive Predictors

In order to compare and estimate the validity of the results obtained with the graph-based model, we explored a Feed-forward Neural Network approach and a simple Graph Convolutional Network.

- FNN: model performing a regression task with three hidden layers of size (512, 1024, 512) using *ReLU* as activation function and regularization techniques as *Dropout*, *Batch normalization* and *Ridge regression*. We obtained good results but worse compared to the other models as we would expect, since the network has a hard time distinguishing various speed peaks in short time intervals.
- GCN: we explored a simple Graph Convolutional Network composed of a single convolutional layer and two linear layers, with the purpose of future days prediction instead of using time windows, finding out no significant advantages and higher error metrics.

4.5 Experiment Result

In the following tables, we report our obtained empirical result. In bold are reported the better results we obtained with respect to the reference model (i.e., STGCN Cheb Yu et al.), having changed the data preprocessing, slightly the model structure and fine-tuned the parameters. In addition, we report plots of the prediction during morning peak and evening rush hours, as shown in Fig. 4 and Fig. 5. It is easy to observe that STGCN is able to capture the trend in a pretty accurate way with respect to the ground truth. While we are achieving pretty the same or sometimes better performances with respect to the basic STGCN model with the PeMSD7 and PeMSD4 datasets, we are experimenting bad results with the dataset MetrLA. We believe this is due to the skewness of the data and high dimensionality of the latter dataset. We are trying to figure out how to improve the overall performance for being able to generalize also on this dataset.

Model	PeMSD7(M)-LA (15/ 30/ 45 min)		
	MAE	MAPE (%)	RMSE
Historical Average	3.64/4.43/5.14	8.32/10.39/12.30	7.07/8.54/9.77
FNN	3.79/ 3.86/ 4.00	9.80/ 10.19/ 10.62	6.46/ 6.81/ 7.07
Simple GCN*	9.94	-	12.93
STGCN Cheb Yu et al.	**2.25**/ 3.03/ 3.57	**5.26**/ **7.33**/ 8.69	**4.04**/ 5.70/ 6.77
Our STGCN Model (Cheb)	2.30/ **2.96**/ **3.38**	5.52/ 7.47/ **8.46**	4.10/ **5.42**/ **6.18**
STAWGNN (state of the art 2021)	1.26/ 2.53/ 2.57	1.66/ 3.43/ 3.53	1.98/ 4.29/ 4.18

Table 1: Performance comparison of different approaches on the dataset PeMSD7

Model	PeMSD4(M)-Bay (15/ 30/ 45 min)		
	MAE	MAPE (%)	RMSE
Historical Average	2.28/2.75/3.18	4.85/6.04/7.13	5.20/6.30/7.22
FNN	2.76/ 3.02/ 3.24	6.82/ 7.05/ 7.14	5.05/ 5.85/ 5.87
Simple GCN*	10.56	-	12.64
STGCN Cheb Yu et al.	1.63/ 2.90/ **2.96**	**1.81**/ 4.17/ 4.27	2.49/ 5.79/ 5.69
Our STGCN Model (Cheb)	**1.50**/ **2.41**/ 3.02	1.95/ **3.15**/ **4.01**	**2.28**/ **3.59**/ **4.49**
MegaCRN (state of the art 2022)	0.78/ 1.52/ 1.40	0.85/ 1.68/ 1.63	1.16/ 2.44/ 2.56

Table 2: Performance comparison of different approaches on the dataset PeMSD4

Model	MetrLA (15/ 30/ 45 min)		
	MAE	MAPE (%)	RMSE
Historical Average	4.94/5.79/6.54	-/-/-	10.93/12.57/13.87
FNN	5.54/ **6.63**/ 7.30	-/ -/ -	10.04/ **11.31**/ 11.75
Simple GCN*	14.67	-	17.48
STGCN Cheb Yu et al.	**2.88**/ 7.62/ 5.74	**3.47**/ **9.57**/ **7.24**	**4.59**/ 12.70/ **9.40**
Our STGCN Model (Cheb)	3.29/ 8.16/ **5.17**	6.64/ 10.35/ 8.73	5.51/ 13.31 /11.02
MegaCRN (state of the art 2022)	1.93/ 4.63/ 3.25	1.99/ 4.85/ 3.53	2.60/ 6.70/ 4.95

Table 3: Performance comparison of different approaches on the dataset MetrLA

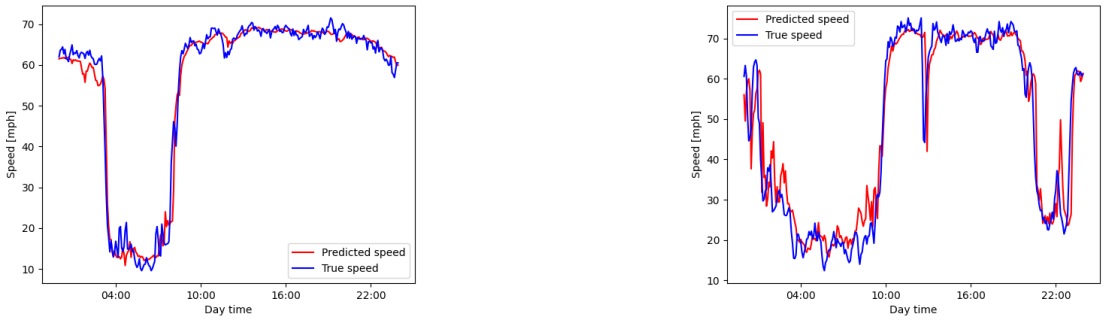


Figure 4: Speed prediction of the dataset PeMSD7

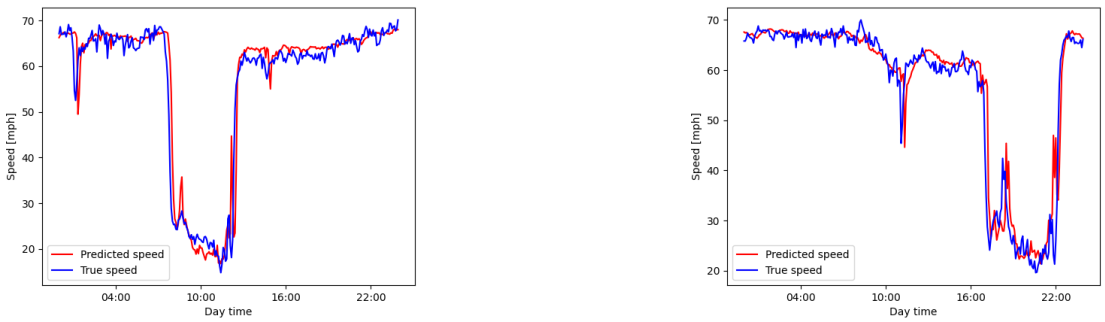


Figure 5: Speed prediction of the dataset PeMSD4

References

[1] David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. In *IEEE Signal Processing Magazine*, 2013.

[2] Jiang et al. Spatio-temporal meta-graph learning for traffic forecasting. 2022.

[3] Seo Y. et al. Structured sequence modeling with graph convolutional recurrent networks. In *https://arxiv.org/pdf/1612.07659.pdf*, 2021.

[4] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[5] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *arXiv preprint*, 2014.

[7] Arlindo Oliveira João Rico, José Barateiro. Graph neural networks for traffic forecasting. In *arXiv:2104.13096*, 2021.

[8] David Leonhardt. Vehicle Crashes, Surging. *New York Times*, 2022.

[9] Mathias Niepert, Mohamed Ahmed, Konstantin Kutzkov. Learning convolutional neural network for graphs. In *Published as a conference paper at ICLR 2017*, 2016.

[10] Michael Defferrard, Xavier Bresson, Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29 (2016)*, 2016.

[11] Brian C. Tefft. American Driving Survey: 2020 – 2021. *AAA foundation*, 2021.

[12] Thomas N. Kipf, Max Welling. Semi-supervised classification with graph convolutional networks. In *Published as a conference paper at ICLR 2017*, 2017.

[13] Yexin Li, Yu Zheng, Huichu Zhang, Lei Chen. Traffic prediction in a bike-sharing system. In *SIGSPATIAL, page 33. ACM*, 2015.

[14] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI), https://www.ijcai.org/proceedings/2018/0505.pdf*, 2018.

Additional: contribution of each member

Our repository is public and is available at this link. Specifically, our contribution in the detail is reported in the following:

- Cristian: creation of the model architecture with the relative code, training of the network, fine tuning of hyper-parameters on validation set, test of the performances for all the 3 datasets, tables comparison, inference on models for plotting predictions.
- Alberto: Initial gathering, comparison and research of useful papers in order to get some result to use as reference for our results, and tried to transform data raw from PeMSD7 dataset into datasets that would fit with our project.
- Simone: implementation of naive predictors as historical average and feed forward neural network for performance comparison, exploration of different prediction approaches with a simple graph convolutional network.