TRAFFIC JAM PREDICTION BASED ON SPATIO-TEMPORAL GRAPH CONVOLUTIONAL NETWORKS

Tianjian Li
Baidu Inc., Beijing
{litianjian}@baidu.com

ABSTRACT

Using Graph Neural Networks in spatio-temporal modeling has proven to be successful in traffic flow prediction. However, there are few attempts to study traffic jams because they most often occur in large cities and in rush hours, resulting in a unbalanced dataset with massive time periods that are not jammed and a few time periods that has traffic jam. The patterns of traffic jams are often daily or weekly periodic. Moreover, these jam patterns has spatial dependencies as well: when one road becomes jammed, its neighbouring roads often jam as well. In our paper, we base our model on [1], changing the graph convolution layer into a graph attention layer, adding a few optimizations to deal with overfitting and oversmoothing, and finally modify the training task from regression to classification to predict the traffic jam status. We evaluate our model on real world jam data drawn from the trajectories in Haidian District, Beijing. Our model outperforms STGCN and other GNN models. By a ablation experiment, we show that our method of adding historic pattern data boosts the performance of our model. The code of our model is at https://github.com/truthbutcher/JamSTGCN.

***K*eywords** Spatio-Temporal Graph Neural Networks · Traffic Forecasting

1 Introduction

Traffic forecasting has become a vital task in nowadays transportation systems. A better forecast of the future traffic flow can substantially boost the performance of ETA retrieval and ranking tasks in route recommendation systems. Among the various traffic flow indicators, jam status plays a crucial role. However, forecasting the specific jam status of a relatively short time interval especially in large cities such as Beijing is still a difficult task, since the jam status manifests both temporal patterns and spatial relations. In this paper, we aim to utilize spatial-temporal graph neural networks to make better predictions of jam status.

There has been various attempts in representation learning in traffic networks. In capturing non spatial dependencies, [2] aims to study a self adapted adjacency matrix. [3] uses Dynamic Time Warping algorithm to model similarities between traffic patterns of different places. [4] and [5] tries to model both long term and short term temporal patterns and dependencies. However, these model's objectives are often forecasting traffic flow based on data of a few hours. Traffic jams often exhibit stronger daily and weekly patterns. For example, traffic jam around recreational places often occur in weekends while in weekdays, places around office buildings are more likely to be jammed with commuters. In order to study these patterns, we based our model on [1], with a few modifications in the model structure to incorporate historical patterns into the forecasting process, which we will explain later in this paper.

We evaluate our model on real life trajectory data from September 2021 to October 2021. We divide each day into 15 minutes and 60 minutes time intervals to generate two different datasets. Jam status of each interval is acquired from trajectory data from Beijing. Moreover, we filtered and clustered a few neighbouring road segments to generate what we call a JamSegement. We then run our model on these JamSegements.

2 Preliminaries

A Graph is denoted as $G = (V, E)$, where V denotes the set of vertices(also named nodes) and E denotes the set of edges. Each vertex has an d-dimensional feature vector. Concatenating the feature vectors of the vertices gives us the feature matrix $X \in \mathbb{R}^{N*d}$. Where N is the number of nodes and d is the number of features. The label associated to each node generates the label matrix $Y \in \mathbb{R}^{N*C}$, where C is the number of different labels.

The adjacency matrix of graph A, whose entry A_{ij} is the weight of the edge from node i to node j. In this paper, edges does not carry weight, therefore every entry of A is 1. The degree matrix D is a diagonal matrix where D_{ii} is the number of edges connected to node i.

2.1 Graph Convolution Networks

The Fourier Transform on graphs is defined as

$$f(x) = U^T x \tag{1}$$

where U is composed by the eigenvectors of the normalized graph laplacian matrix L.

$$L = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{2}$$

Therefore the convolution on graphs is defined as

$$g_\theta * (x) = U g_\theta U^T x \tag{3}$$

Where g_θ is a diagonal matrix regarded as the filter. Here $U^T x$ first transforms our graph signal into the spectral domain, then we apply the filter g_θ to it. We then transform the result back to the spatial domain by multiplying U. Evaluating (3) is computationally expensive, as it requires us to compute the eigendecomposition of the normalized graph laplacian matrix. [6] uses a first order Chebyshev polynomial to approximate the graph convolution, defining a Graph Convolution Network(GCN) layer as

$$H^{(l+1)} = (I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) H^{(l)} W^{(l+1)} \tag{4}$$

where $H^{(l)}$ denotes the output of the lth layer, and $H^{(0)} = X$, the input feature matrix. In practice, in order to avoid gradient instabilities, GCN is applied with a renormalization trick proposed in [6]. We precompute $\tilde{A} = A + I_N$ and $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$ where $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. The resulting GCN layer update equation would be

$$H^{(l+1)} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l+1)} \tag{5}$$

2.2 Graph Attention Networks

[7] proposes Graph Attention Networks (GAT), which is now widely used to produce SOTA results in node feature classification tasks. Instead of using a convolution filter to aggregate the neighbouring node features, GAT assigns each neighbouring node an "attention weight" that can be learned. The attention weight is calculated by

$$a_{ij}^l = \frac{exp(LeakyReLU(a^T [W^{(l)} x_i^{(l)} || W^{(l)} x_j^{(l)}]))}{\sum_{r \in N(v_i)} exp(LeakyReLU(a^T [W^{(k)} x_i^{(l)} || W^l x_{(j)}^{(l)}]))} \tag{6}$$

and the update rule of each GAT layer is

$$x_i^{(l+1)} = \sigma\left(\sum_{v_i \in N(v_i)} a_{ij}^{(l)} W^{(k)} x_j^{(l)} \right) \tag{7}$$

2.3 Spatio-Temporal Graph Convolutional Networks

Spatio-Temporal Graph Convolutional Networks (STGCN) was proposed in [1] to tackle with traffic flow prediction. In this paper, we use the framework of STGCN because it is a simple 2-layer model which is easy to modify and add utilize additional optimizing tricks.

Each Spatio-Temporal conv Layer is composed of two temporal conv layer with a spatio conv layer between them. Figure 1 except the additional Spatial Conv Layer in the left part is the original model structure. The "sandwich" like structure allows the model to aggregate temporal patterns, pass them through edges in the graph, and aggregate neighbouring temporal patterns in the last temporal conv layer. The deeper a graph neural network is, the more easily the features of the nodes in a graph becomes very similar. Because each layer of GNN aggregates neighbouring features, after the features gets aggregated for a substantial amount of layers, the representation of each node becomes the sum of the feature of its connected component, a negative outcome which we call oversmoothing. In the next section, we will directly show the modifications we have made to deal with oversmoothing.

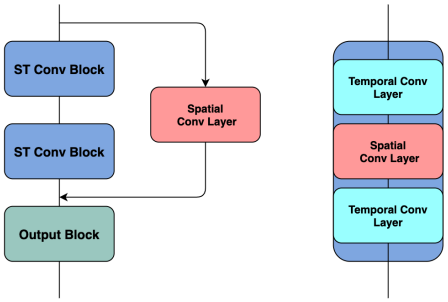

Figure 1: Model of modified STGCN

3 Methodology

Forecasting traffic jams raises problems different from forecasting traffic jams. In [3] and [1], the primary intention of those models was to forecast traffic flow, which we have more than enough data to manipulate. However, traffic jams(especially serious ones) often occur in business areas during rush hours. Most of the time we won't experience any jam at all in suburban areas. Therefore the jam data we have is astronomically less than data with smooth traffic. To learn the spatial and temporal patterns of traffic jams, we filtered through one of Beijing's most crowded business area - The XiErQi area where large technology companies(Baidu, Xiaomi, Kwai) locate, and selected few road segments to form our graph. We also use other techniques than sampling used to tackle with the disproportional jam data, which we will addressed later in this paper.

The input of our model contains two parts, one is the collection of jam status before our query time, one is the collection of jam status at the same time for the past few days. For example, if the desired time for prediction is November 30, 2021 at 8P.M., if we fix the hyperparameter of the number of entries T to be 12, the first part consists of jam status from Nov 30, 2021 from 8A.M. to 7P.M., one integer for each hour. The second part of the input consists of the jam status of 8P.M. on November 18 to November 29, one integer for that specific hour of each day. We fix the length of the two part to be the same to concatenate them in the final output block. We only feed the first part of the input to the ST Conv blocks, the second part is only goes through a Spatial Conv Layer.

Inside each ST Conv Block, only the input $X \in \mathbb{R}^{B,T,N,C}$ first goes through a Temporal Conv Layer, here B is the size of batches, N is the number of nodes and C denotes the number of channels. 1-D convolutional network is applied to the input to extract temporal patterns. Then the input goes through a Spatial Conv Layer, where a chosen graph operation layer is applied to the input. In the original STGCN paper, the author uses ChebNet[8] and GCN[6]. In our experiments, we used GCN[6], GAT[7], and GraphSAGE[9] as our choice of graph operations.

3.1 Incorporation of historical patterns

Traffic jam patterns often follows a daily routine. In XiErQi area, where a lot of technological companies locate, traffic jams occur in the morning before 10A.M., when most companies begin their worktime and at night starting from 7P.M., when a lot of people return from work. However, only looking at a few hours of jam status in the past might not be clear enough for the model to learn such patterns and routines. Therefore we directly feed the model data from the past days at the same time, to let the model directly learn the pattern. Experiment results show that the additional knowledge from the past few days are crucial in predicting traffic status.

3.2 Skip Connections

Skip connections was introduced to the area in image classification in [10], in order to learn identity mappings. Here skip connections is added in each Spatial Conv Layer. As shown in Figure 2, the input is directly added to the output of the graph operation, followed by an activation function. Skip Connections also helps against oversmoothing in graphs since the "unsmoothed" features is directly added to the output.

3.3 Weighted Cross Entropy

The cross entropy H of a distribution q relative to a distribution p is defined as follow:

$$H(p, q) = -E_p[\log q] \tag{8}$$

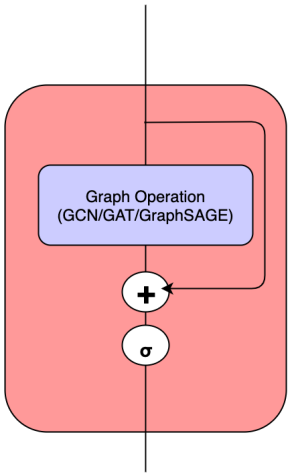

Figure 2: Skip Connection

where $E_p[\cdot]$ is the expected value operator under distribution p. If p is the distribution of a discrete random variable, then the cross entropy is calculated by

$$H(p, q) = -\sum_{x \in P} p(x) \log q(x) \tag{9}$$

where P is the set of all possible values of the discrete random variable. However, (9) assigns importance to labels based on their occurrence. If smoothed traffic, labeled by 1 appears too much, the model would only learn about the pattern of no traffic jam, which is ubiquitous. In order for the model to learn patterns of traffic jam better, we use a set of hyperparameters to weight the different labels. After weighting, the cross entropy loss becomes

$$H(p, q) = -\sum_{x \in P} W(x) P(x) \log q(x) \tag{10}$$

where $W(x)$ is the weight assigned to the label x.

4 Experiments

5 Conclusion

Acknowledgments

References

[1] 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)*, 2018.

[2] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR '18)*, 2018.

[3] Mengzhang Li and Zhanxing Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4189–4196, May 2021.

[4] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):914–921, Apr. 2020.

[5] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):922–929, Jul. 2019.

[6] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[9] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.