DE-NOISING IS ALL YOU NEED: AN SVD-BASED METHOD TO ENHANCE THE PERFORMANCE OF GRAPH NEURAL NETWORKS IN TRAFFIC FLOW PREDICTION

Yangyang Qi [a], Xiaoyang Xin [a], Zesheng Cheng [a], Tiankuan Wang [b], Bangyang Wei [c] & Zhenkuan Pan [a]

Abstract

High-quality traffic flow data may provide planners with a basis for designing road capacity, pavement and intersection control, etc., thus assisting in the construction of a more rational traffic network. In this paper, a new approach to improve the performance of traffic flow prediction models is proposed. Based on the researches of graph neural networks by spectral graph and graph signal processing theory, this approach combines low-pass filter and de-nosing process with graph convolutional neural networks. In particular, this study applies k-order Singular Value Decomposition process as de-noising method and simply added it to the Spatial Temporal Graph Convolutional Network model, an earlier application of graph convolutional module in traffic flow prediction tasks. The proposed model, tested on several public datasets, performs better than or similar to the state-of-the-art baseline model. Therefore, it is believed that it may provide a different perspective for discussing the application of graph neural networks on the traffic flow prediction tasks.

Keywords: Graph Convolutional Network, Low-pass Filter, De-noising, Traffic Flow Prediction.

1. Introduction

Since the beginning of the 21st century, large and medium cities around the world have started to deploy Intelligent Transport System (ITS), which aims to provide safe, convenient, and comfortable transport and effectively addresses issues about traffic congestion and environmental pollution [1,2,3]. With the development of ITS, accurate collection of spatial-temporal data is of great significance, which has led to the relevant topic being very actively discussed in industrial and scientific fields. Traffic flow data is undoubtedly one of the critical and important among them. High-quality traffic flow data may provide planners with a basis for designing road capacity, pavement and intersection control, etc., thus assisting in the construction of a more rational traffic network [4,5,6]. In practice, relevant data is often collected via road sensors or video surveillance (as shown in Figure 1 below) set up by

governments or authorities. These devices are budget costing, limited to coverage area, also susceptible to weather and road conditions [7]. Hence, with the development of information technology, practitioners tend to apply machine learning methods as an alternative to forecast or simulate inaccessible traffic flow data [8,9,10].

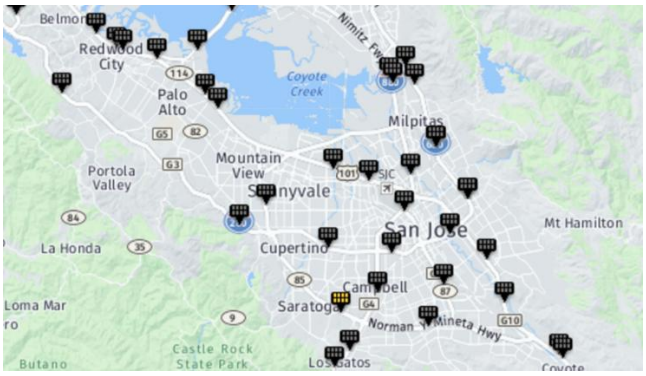

Figure 1: Traffic Sensors around San Jose, USA [55]

Traffic flow data is usually considered as a class of unstructured data that cannot be easily expressed in Euclidean space [11,12,13]. Therefore, in traffic flow prediction tasks, the graph neural network (GNN) structure is currently the most commonly used. That is because it to some extent solves the problem that traditional deep learning cannot process unstructured data [14]. Based on GNN, researchers have developed models that may predict the traffic flow of target nodes in the network at any time slices by combining with temporal neural networks [15]. Since then, more outcomes from deep learning research, such as attention module [16,17], have been added to GNN-based traffic flow prediction model. Although some results have been achieved, they have simultaneously made the entire model structure more redundant and complex. Some models even perform worse when more complicated modules are employed.

Therefore, instead of stacking advanced but complex deep learning modules, it is necessary to find an elegant and effective method to enhance the performance of GNN-based model in traffic flow prediction tasks. This study proposes a new improvement method based on spectral graph and graph signal processing theory, which is combining the GCN-based model with de-noising process. In particular, the method outperforms the state-of-the-art model on public datasets only by adding a simple k-order singular vector decomposition (k-SVD) de-nosing process after each training epoch of STGCN. Meanwhile, the model is simple in structure, fast in training, and stable and effective after many tests. Therefore, it is believed that the method may provide a different perspective for discussing the application of graph neural networks to the traffic flow prediction tasks. Theoretical basis of the method and related experimental results will be provided in the rest of the manuscript.

2. Related Works

This section presents the related theory and works of GNN-based model applied in

traffic flow prediction tasks as well as analysis and conjectures.

2.1 Traffic Flow Prediction and Earlier Spatial-Temporal Model

In relevant studies, traffic flow prediction is considered to be a typical spatial-temporal data prediction problem [18]. The traffic conditions of the adjacent road sections as well as the past period of time at the target section itself have an impact on the flow data of the target road section at the target time period. In other words, spatial and temporal correlations are key factors to consider when making predictions [19,20]. Since processing time series has been a hot topic in industry and academia for a long time, models for extracting temporal features are currently more advanced. Simple examples include Temporal Convolution Network (T-CN) [21] and Recurrent Neural Network (RNN) [22]. While the discussion on capturing spatial features, on the other hand, started later. Different from data such as speech, images or text, which have a simple sequence or grid structure, traffic networks or graphs abstracted from traffic networks, have a more complex topological structure. Such structures are generally irregular and disordered, making correlation, especially spatial correlations between traffic flows significantly difficult to express in a structured method [23].

The proposal of graph neural network (GNN) [24] model has largely contributed to the solution of the above problem. Essentially, a GNN is a connectivity model that acquires dependencies by learning how message is passing between nodes in the graph. In this way, the target node interacts with its neighbors at any depth in the graph with information to update its state and represent relevant information [25]. Since GNN learns in a way that message is passing through each node separately, there is no need to arrange the features in a structured way, thus allowing for more convenient processing of complex unstructured data [25,26,27,28]. Based on GNN model, graph-based neural network models adapted to different scenarios and data types have been rapidly applied to different tasks [29,30,31,32,33]. In recent year, it has also been introduced and have made great progress in the field of traffic flow prediction.

Among these GNN-based models that have been applied to traffic flow prediction, the Spatial-Temporal Graph Convolutional Network (STGCN) model is one of the earlier and more fundamental one [34]. It stacks two T-CN modules (including Casual Convolution and Residual Connections) and a Graph Convolutional Network module (GCN) to extract the temporal and spatial correlations of traffic flow, respectively. The model has a simple structure and is fast to train. Meanwhile, its performance is well and stable in various public datasets. Similar models include Temporal-Graph Convolutional Network (T-GCN) [35] that replace the T-CN model with a Gated Recurrent Unit (GRU) [36] module.

Early models themselves had some shortcomings, for example, they used only a single layer of unweighted binary graph convolution to extract spatial correlations.

That is, these models assume that traffic flow of target node in a traffic network is only spatially correlated with its one-hop neighboring nodes. This assumption is clearly not sufficient enough. Overall, early models provide a reliable path for the improvement and optimization of subsequent models.

2.2 Improvement and Optimization of GNN-based Models

From the previous section, it can be concluded that the basic process of developing a traffic flow prediction model is: firstly, the traffic network and flow data are represented as a set of graph data; then, the temporal features and spatial features are captured by temporal and spatial neural network modules, respectively; finally, the extracted features are fused. Therefore, the improvement of the model may also be performed in three aspects, which are, improving the graph representation method, spatial module or temporal module. According to the topic of this study, only the first two aspects are discussed in details.

Firstly, is to change the graph representation method from using traditional binary adjacent matrix to using a learnable distance parameter. This allows message passing not only between one-hop neighbors in the graph, but also between multi-hop neighbors with close distances. Representative methods in this category include Diffusion Convolutional Recurrent Neural Network (DCRNN) [37] and ST-MetaNet [38].

Further, is the application of learnable adaptive graph representations methods. Instead of evaluating the message passing between vertices in terms of whether they are connected or not, the optimal message passing method is learned introducing the graph learning layer and external knowledge. That allows message passing between related vertices with arbitrary number of hops. Representative methods in this category include Spatial-Temporal Graph Neural Network (STGNN) [39], Graph WaveNet [40] and Multi-Task Graph Neural Network (MTGNN) [41].

The most important improvement regarding spatial modules is the introduction of attention mechanism to GCN. This category of model is concerned with extracting the features of information passing within a one-hop neighborhood more accurately. Representative methods in this category include Attention-based Spatial Temporal Graph Convolutional Network (ASTGCN) [42] and Graph Multi-Attention Network for Traffic Prediction (GAMN) [43].

Other representative models include Dynamic Graph Convolutional Recurrent Network (DGCRN) [44] model that can dynamically generate hyper-networks before network training.

2.3 Analysis and Research Gaps

Plenty of experiments have proved that for the traffic flow prediction tasks, models that considering the message passing between multi-hop neighbors, such as MTGNN

and the DCRNN, outperform the models which considering to capture the message passing patterns between one-hop neighbors more accurately, such as ASTGCN and GMAN. DCRNN model, the top-ranked model among many public traffic flow prediction tasks, even discusses the case of steady state after infinite message passing through the graph. The core formula is the p-step truncation for this case. Therefore, although influence by relevant factors such as temporal modules, a reasonable conjecture is that considering the message passing between multi-hop neighbors may have a more significant role in the traffic flow prediction tasks than capturing the message passing patterns between single-hop nodes more accurately. In the rest of the manuscript, the earlier STGCN model will be improved according to this conjecture.

3. Graph Spectral Theory and Graph Signal Processing

This section provides a brief introduction of graph spectral theory and graph signal processing.

3.1 Preliminaries

In mathematics [45], an unweighted graph A is usually represented by $G_A = \{V_A, E_A\}$, where V_A is the set of vertices and E_A is the set of edges between vertices. In traffic flow prediction task, the traffic network is generally abstracted into a simple graph which is undirected and unweighted. Since graphs data are unstructured, it is difficult to have a structured representation that reflects the entire information in the graph. Parts of the key information on the graph are represented in the form of matrices as follows.

Adjacent matrix A, which A_{ij} represent the weight (1 in unweighted graph) of the edge between upstream vertex i and downstream vertex j.

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{nn} \end{bmatrix}$$

On the top of adjacent matrix, degree matrix D is defined. D is a diagonal matrix where D_{ii} represent the degree of vertex i, which is the sum of the edges' weights connected to the target vertex.

$$D = \begin{bmatrix} D_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_{nn} \end{bmatrix}$$

In addition, the adjacent matrix A is not only a representation of the graph topology, but also a one-hop operator of the graph signal. It is a transformation that operates only within the one-hop neighborhood of a vertex. Assume that the signal matrix consisting of the vectors of all nodes in a graph is x. Then the physical meaning of the transformation Ax is that the signals and related information of the upstream vertex i

are transmitted to the downstream vertex j along the edge A_{ij}. This transform is the basis of message passing process in GNN models. Similar one-hop operators include the Laplace operator L = D – A. Properties of these operators will be described in detail in subsequent sections.

3.2 Spectral Graph Theory and Graph Signal Processing

With the help of one-hop operator adjacent matrix A, the GNN model can realize the message passing between vertices on the graph. The tasks that GNN can accomplish are categorized into node-level tasks (including node classification, regression and clustering), edge-level tasks (including edge classification and link prediction) and graph tasks (including graph classification, regression and matching). The most common traffic flow prediction task can be viewed as a node regression task. It predicts the traffic flow at each node by learning the weights of the message passing on each edge through GNN model. Similar to convolutional neural networks (CNN), if the commonalities in the message passing between each hop can be extracted and learned, the size of the network as well as the training time can be greatly reduced. However, unlike CNN, due to the fact that graph data is unstructured, the operation of convolution could not be performed directly in the graph spatial domain. The initial graph convolutional neural networks (GCN) model was developed in the spectral domain of the graph.

Spectral Graph Theory suggests that any graph signal x can be decomposed into a number of base signals by applying one-hop operator Z, where $Zx = Zx_1 + Zx_2 + \ldots + Zx_n$. This decomposed transforms the graph signal from the graph spatial domain to spectral domain with base of $[x_1, x_2, \ldots, x_n]$. Thereby, it effectively avoids the problem of inconveniently processing unstructured data in the graph spatial domain. In particular, if the one-hop operator is diagonal, the base signal is the set of eigenvectors of the operator. Applying eigenvalue decomposition to the operator $Z = U\Lambda U^{-1}$, then $x^{spe} = U^{-1}x^{spa}$ transforms the graph signal from spatial domain to spectral domain and $x^{spa} = Ux^{spe}$ transforms the signal back to spatial domain. Figure 2 below is an example of a simple graph signal.

Figure 2(a): Original graph

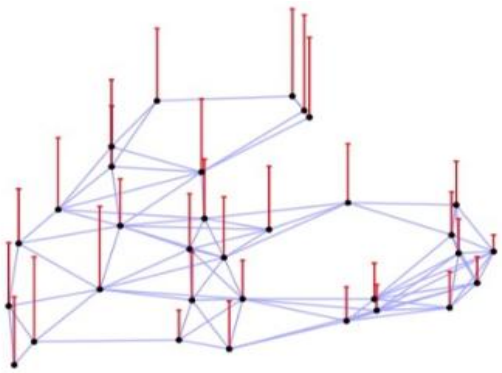

Figure 2(b): Graph signal

Figure 2: Graph and corresponding graph signal

The primitive GCN model applies Laplace operator L. The model can be expressed as:

$$H^{n+1} = \sigma(Ug(\Lambda)U^{-1}H^n w) \tag{1}$$

GCN model will learn to process the signal in the spectral domain g(Λ) and the weight of the convolution layer w. By the operation of transforming the signal to spectral domain, processing the signal in spectral domain and then transforming it back to spatial domain, GCN extracts the commonality of message passing between neighboring vertex at each hop. This results in a significant reduction in the training cost of the graph model.

It is worth noting that the quadratic form of the Laplace matrix is (Please refer to Section 7: Formula Derivation for more details):

$$\mathrm{x}^T\mathrm{Lx} = \sum_{(i,j)\in E} A_{ij}(x_i - x_j)^T(x_i - x_j) \tag{2}$$

If x is an eigenvector of L and λ is the corresponding eigenvalue, then:

$$\mathrm{x}^T\mathrm{Lx} = \mathrm{x}^T\lambda\mathrm{x} = \lambda\mathrm{x}^T\mathrm{x} = \lambda = \sum_{(i,j)\in E} A_{ij}(x_i - x_j)^T(x_i - x_j) \tag{3}$$

The quadratic form above indicates that the eigenvalue of the Laplace operator can be considered as a weighted sum of Euclidian distances between a vertex and its one-hop neighbors, which is the difference between their attributes. The larger the difference, the larger the eigenvalue will be accordingly. Similar to signal processing, graph signal processing suggests that the eigenvalues can be applied as the frequencies of the graph signal in the spectral domain. For Laplace operator, smaller eigenvalues correspond to lower frequencies, which is the components of the graph single that are more similar to its surrounding nodes and vice versa.

Moreover, the quadratic form of the one-hop operator adjacent matrix A is (Please refer to Section 7: Formula Derivation for more details):

$$\mathrm{x}^T\mathrm{Ax} = 2\sum_{(i,j)\in E} A_{ij}x_i x_j \tag{4}$$

In contrast to the Laplace operator, the eigenvalues of the adjacent matrix are related to the inner product of the graph signals. The closer the node features are (low frequency), the larger the corresponding eigenvalues are. Because its physical meaning is not clear enough and there are no extreme points, it is not commonly used

in such models such as GCN. In the formula of GCN model, the normalized form of the above two and the adjacent matrix with self-loop ($\widetilde{\mathrm{A}} = A + I$) are more commonly used. The characteristic of their corresponding eigenvalues remains unchanged.

However, due to the time complexity of eigenvalue decomposition is $O(n^3)$, the computation time is still unacceptable with the size of graph increase. Therefore, modern GCN models mostly replace the eigenvalue decomposition of Laplace operator by polynomial approximation. Currently the most common model is Cheb-net, which applies the first order Chebyshev polynomials as an approximation. The model can be represented as:

$$H^{n+1} = \sigma((D + I)^{-1/2}(A + I)(D + I)^{-1/2}H^n w) \tag{4}$$

The Cheb-net applies an adjacent matrix with self-loops renormalized by the degree matrix to represent the message passing process of the graph signal. This replaces the training processing in spectral domain, which only the weights w of each convolutional layer required to be trained.

From the spatial domain, the simplified graph convolution directly applies the adjacent matrix with self-loops to pass message. While from the spectral domain, since:

$$(D + I)^{-\frac{1}{2}}(A + I)(D + I)^{-\frac{1}{2}} = \mathrm{V}(\mathrm{I} - \widehat{\Lambda})V^T \ (\widehat{\lambda_i} \in [0,2) \ for \ \widehat{\lambda_i} \ in \ \widehat{\Lambda}) \tag{5}$$

In the equation above, $\widehat{\Lambda}$ it the eigenvalue matrix of normalized Laplace operator. Since $\widehat{\lambda_i} \in [0,2)$ (Please refer to Section 7: Formula Derivation for more details), therefore, for a graph signal, the simplified single-layer graph convolution is equivalent to a linear low-pass filter, with filter function (1 - $\widehat{\lambda_i}$). The higher the frequency of the graph signal and the higher the value of $\widehat{\lambda_i}$, the more dramatic the attenuation after passing through this filter. If p-layer graph convolution modules are stacked, the filter function becomes $(1 - \widehat{\lambda_i})^p$, which in essence still a low-pass filter.

3.3 K-order Singular Vector Decomposition (k-SVD)

Singular value decomposition is a matrix decomposition method available for matrices of arbitrary sizes. It decomposes the original matrix into left singular matrix, singular value matrix and right singular matrix (Eq. 6 & 7).

$$A^T A = (\mathrm{U}\Sigma V^T)^T \mathrm{U}\Sigma V^T = V\Sigma^T \Sigma V^T \tag{6}$$
$$AA^T = \mathrm{U}\Sigma V^T (\mathrm{U}\Sigma V^T)^T = U\Sigma\Sigma^T U^T \tag{7}$$

The method of reducing the original matrix by taking the top k (k << the size of original matrix) singular values and their corresponding singular vectors, is named k-SVD method. This method is mostly used in the field of principal component analysis, de-nosing process, low rank decomposition and recommendation engines.

4. Methodologies

This section describes the method to improve STGCN model based on de-noising process and the two models proposed in this study.

4.1 Linear Low Pass Filter to Ideal Low Pass Filter

As it is described above, it is clear that the first-order Cheb-net is equivalent to a linear low pass filter with a filter function of (1 - $\widehat{\lambda}_i$) for graph signals. Further, the p^{th} order Cheb-net that can describe the message passing between p-hop neighbors is then equivalent to a low-pass filter with filter function $(1 - \widehat{\lambda}_i)^p$. Following Figure 3 illustrates the image of the corresponding filter function when p takes different values.

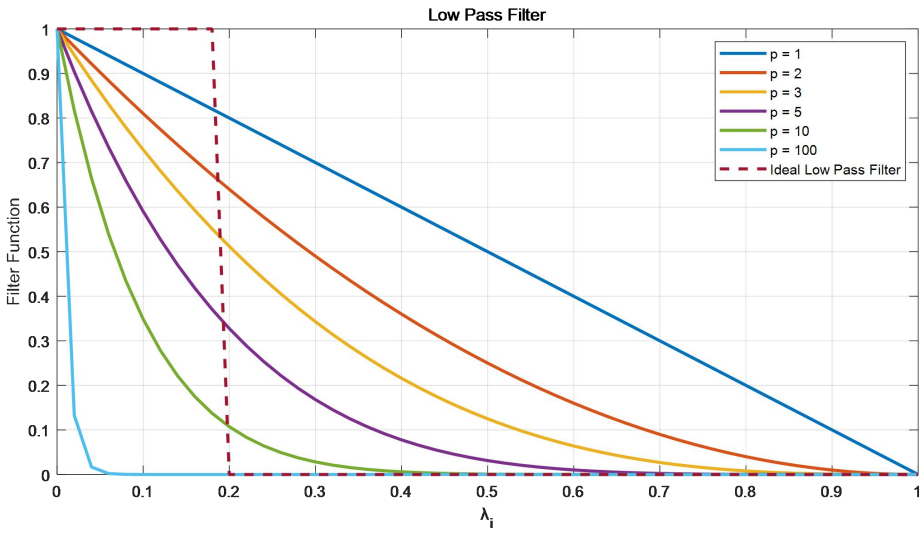

Figure 3: Filter function with different p and ideal low pass filter

As can be seen from the Figure 3, the larger the value of p, the lower frequency that signals pass through the filter will be attenuated close to 0. In fact, if the message is passed infinitely many times in the graph according certain rules, eventually only the component with a frequency of 0 can pass through the filter. At the point, the information will be same for all vertices in spatial domain. In order to avoid over smooth, it is necessary to truncate p to retain some of the differential information.

In summary, this study concludes that the message passing process of the p^{th}-order Cheb-net can be approximated by an ideal low pass filter as shown in the Figure 3. This filter allows lower frequency signals to pass through intact while higher frequency signals attenuate to 0. This study designs a graph convolution module as shown in the figure 4 below following this idea.

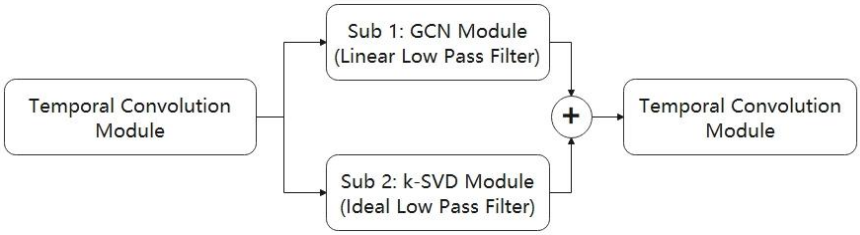

Figure 4: Proposed GCN module

This module has two sub-modules. Sub-module 1 is the original 1-order Cheb-net. Sub-module 2 is the ideal low-pass filter applying k-SVD. In order to maintain the same structure with Cheb-net, the adjacent matrix with self-loop ($\widetilde{\mathrm{A}}$) is applies as the one-hop operator. k-SVD($\widetilde{\mathrm{A}}$) can be used as an ideal low-pass filter because it applies the product of the top-k singular values and singular vectors in ($\widetilde{\mathrm{A}}$) to represent the original matrix. In spectral graph theory, it equivalent to represent the original graph signal only by the k-lowest frequency components while discarding those with higher frequency, thus accomplishing the ideal low-pass filtering of the graph. The two sub-modules are connected in parallel with a similar method to residual connection and assigned trainable weights respectively. Then, the result is multiplied with the graph signal to form a completed GCN module. With this module, the study designed Model I in Section 5 below.

4.2 De-nosing is All You Need

However, there are still two issues for the model above. One is that the model adds a series of trainable parameters in graph convolutional model. This may result in more computational resources to be consumed by the GCN model, which is already expensive to train. Second, further processing may need for the model to get competitive performance on the traffic flow prediction tasks. To address these two issues, this study designed Model II which will be further discussed in Section 5. Model II replicates the initial structure of the STGCN exactly, only adding a k-SVD process on the entire training results at end of each training epoch, before the loss function is computed.

This study suggests that the reasons for the elegance and effectiveness of this simple process can be concluded in the three aspects below.

First of all, STGCN consists of two temporal convolution modules (TCN) and one spatial convolution module (GCN). Each row of its output represents the traffic of all nodes on the graph at a time slot, while each column represents the traffic of a single node on the graph at all time slot. Accordingly, the SVD process decomposes the original matrix into a left singular matrix and a right singular matrix, which corresponds to the row (spatial domain) and columns (temporal domain) to the original matrix, respectively. For spatial information, the output of STGCN model can be approximated as $\widetilde{\mathrm{A}}x$, i.e., message passing between target vertex and its one-hop neighbors. The quadratic form of the operator $(\widetilde{\mathrm{A}}x)^{\mathrm{T}}(\widetilde{\mathrm{A}}x)$ is (Please refer to Section 7:

Formula Derivation for more details) :

$$\mathbf{x}^T\left(\widetilde{\mathrm{A}}\boldsymbol{x}\right)^T\left(\widetilde{\mathrm{A}}\boldsymbol{x}\right)\boldsymbol{x} = \sum_{k=1}^{i} x_k^2 \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (x_i + \sum_{j=1}^{n} a_{1j}x_j)^{\mathbf{2}} \tag{8}$$

Since the left singular matrix decomposition is performed as:

$$[\left(\widetilde{\mathrm{A}}\boldsymbol{x}\right)^T\left(\widetilde{\mathrm{A}}\boldsymbol{x}\right)]\boldsymbol{x} = \sigma^2 \tag{9}$$

If x is a singular vector of $\widetilde{\mathrm{A}}x$ and σ is the corresponding singular value, then:

$$\sigma = \sqrt{1 + 2\sum_{i,j=1}^{n} a_{ij}x_i x_j + (\sum_{i,j=1}^{n} a_{ij}x_j)^2} \tag{10}$$

Although it is not as straightforward as the orthogonal decomposition results of the Laplace and adjacent matrix operators, it is still possible to conclude that the more similar the neighbor vertices are to each other, the larger the corresponding singular values and the lower the frequency of the graph signal. Thus, k-SVD($\widetilde{\mathrm{A}}x$) in spatial domain is equivalent to ideal low-pass filtering of the graph signal.

For temporal information, similar to spatial information, k-SVD can preserve the main components of the temporal information and remove a series of low-intensity components added by data processing and training process. In particular, the time series prediction process generally predicts unknown data from known data, and then predicts subsequent unknown data by the predicted unknown data. In this process, noise (error) accumulates. Therefore, the process of retaining principal components is similar to a noise reduction process. Since temporal module is not the main content of this manuscript, it will not be further discussed. Please refer to [46-48] for more details.

Meanwhile, researchers found that for graph signals, majority of information is also concentrated in the lower frequency components, while noise tends to be concentrated in the higher frequency components. Therefore, k-svd process added after training epoch plays a role of de-noising for both spatial and temporal dimensions. This manuscript is finally named as “De-nosing is all you need”.

In addition, since STGCN itself is a relatively simple model. Meanwhile, the time complexity of k-SVD (without participating in the training process) is $O(kN^2)$ and experiments prove that $k << N$ (k is taken as 3 in the experiments), the extra computational resource consumption of the model is acceptable.

5. Experiments and Analysis

This chapter presents experiments and related data, model, results and analysis of

model performance in traffic flow prediction tasks.

5.1 Experimental Preparation

- **Problem Definition**

The traffic flow prediction task can be defined as: give all kinds of the historical measurements of all the nodes on the traffic network over past t time slices to predict future traffic flow sequences $\mathbf{Y} = (\mathbf{y^1}, \mathbf{y^2}, \ldots, \mathbf{y^N}) \in R^{N*p}$ of all the nodes on the whole traffic network over the next p time slices, where $\mathbf{y^i} = (y^i_{t+1}, y^i_{t+2}, \ldots, y^i_{t+p}) \in R^p$ denotes the future traffic flow of node I from t+1.

- **Dataset**

Three datasets shown in the following table I are applied to the experiments in this section.

Table 1: Datasets

Dataset	No. of Nodes	No. of Time Slices	Description
PEMSBAY [49]	325	52116	Bay Area, the state of California
PEMSD7M [50]	228	12672	District 7, the state of California
METR-LA [51]	207	34272	Los Angeles Metropolitan

The data is divided into training set, validation set and test set in the ratio of 7:1.5:1.5. Since it involves time series data analysis, the division is done by taking successive values according to the time series. The model extracts the relevant information from the past 3, 6 and 12 time slices and accordingly predicts the future traffic flow at the 3^{rd}, 6^{th} and 12^{th} time slices respectively. Relevant models are trained and tested using the same dataset and the average of 10 times tests were taken for the final evaluation.

- **Experimental Environment**

All the experiments in this study are conducted on a server with a single Intel Xeon W-2133@3.6Hz CPU and one 32 RAM NVIDIA V100 GPU.

- **Models**

As mentioned in the previous section, two models are developed in this study as follows:

Model I: replace the graph convolution module of STGCN with the parallel graph convolution module described in subsection 3.1
Model II: Add the k-SVD process after each training epoch of STGCN, with k taking the value of 3.

Since Model I only has an effect on spatial feature extraction, it is equivalent to the ablation experiment of Model II. The rest of the structure is kept exactly the same as STGCN. The reason for choosing STGCN as the base model for modification is because the modules are simpler. As an earlier model of applying GCN to traffic flow

prediction tasks, STGCN model lacks competitiveness gradually in the process of updating the related technology. Therefore, it would be more sufficient to illustrate the effectiveness of the method proposed in the study if the simple process of k-SVD can achieve a performance better than the models with more complex structures.

- **Baseline**

The following six models: STGCN, ASTGCN, GMAN, DCRNN, MTGNN, DGCRN are used as baseline models in this study. Their underlying theory associated with these models has been briefly described in the related work section. It is worth to note that, related studies show that the state-of-the-art model in traffic flow predication tasks should be STGM model. However, the model is less stable. In this study, after several repetitive experiments, it is not able to get acceptable results. So, STGM are not discussed in this manuscript.

- **Evaluation Indicators:**

Following three evaluation indicators are used in this study to assess the performance of the model:

Root Mean Square: $\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{n}(h(x_i) - y_i)^2}$

Mean Absolute Error: $\mathrm{MAE} = \frac{1}{\mathrm{m}}\sum_{i=1}^{n}|h(x_i) - y_i|$

Mean Absolute Percentage Error: $\mathrm{MAPE} = \frac{100}{\mathrm{m}}\sum_{i=1}^{n}\left|\frac{h(x_i)-y_i}{y_i}\right|$

5.2 Experiment Results

Table 2 blow presents the testing result of the six baseline models and the proposed two methods (Model I and Model II).

Table 2: Experiment Results

Datasets	Model	Time Slice								
		15 min			30min			60min		
		RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
PEMSBAY	STGCN	3.334	1.671	3.12%	4.192	2.037	4.04%	5.378	2.616	5.21%
	ASTGCN	3.157	1.505	3.27%	4.116	1.975	4.04%	4.970	2.212	4.85%
	GMAN	3.123	1.580	4.56%	3.936	1.946	4.78%	4.480	2.061	5.36%
	DCRNN	2.934	1.431	3.15%	3.956	1.834	4.26%	4.853	2.134	5.21%
	MTGNN	2.806	1.327	2.93%	3.750	1.646	3.83%	4.475	1.937	4.68%
	DGCRN	**2.685**	**1.285**	**2.69%**	3.638	1.599	**3.61%**	4.388	**1.883**	4.52%
	Model I	3.173	1.760	3.79%	3.718	2.019	4.45%	4.861	2.52	5.19%
	Model II	2.791	1.402	2.82%	**3.530**	**1.547**	3.83%	**4.387**	1.889	**4.47%**
PEMSD7M	STGCN	4.264	2.406	5.36%	5.598	3.142	7.02%	6.789	3.781	8.92%
	ASTGCN	4.357	2.520	5.87%	5.516	2.988	7.69%	6.593	3.601	9.55%
	GMAN	5.625	2.903	7.35%	6.183	3.142	7.88%	7.992	3.996	10.01%

	DCRNN	4.433	2.341	5.58%	5.733	2.961	7.45%	7.226	3.681	9.84%
	MTGNN	4.032	2.200	**5.05%**	5.421	2.732	6.81%	6.550	**3.215**	8.27%
	DGCRN	4.073	2.208	5.21%	5.443	2.769	6.79%	6.733	3.325	8.56%
	Model I	4.171	2.401	5.25%	5.417	2.911	6.95%	6.669	3.581	8.68%
	Model II	**3.972**	**2.134**	5.06%	**4.974**	**2.736**	**6.46%**	**6.320**	3.361	**8.40%**
METR-LA	STGCN	6.272	3.545	9.04%	8.157	4.122	10.72%	9.503	4.621	12.52%
	ASTGCN	6.268	3.621	8.70%	7.555	4.179	10.31%	9.993	4.757	13.11%
	GMAN	8.801	5.592	10.00%	9.276	5.965	10.97%	10.246	7.166	12.79%
	DCRNN	6.292	3.542	8.72%	7.508	4.113	10.35%	9.66	4.756	12.30%
	MTGNN	6.244	3.241	9.03%	7.446	3.680	10.59%	9.051	4.220	12.18%
	DGCRN	**6.045**	**3.146**	**7.97%**	7.291	**3.480**	9.68%	8.693	**4.133**	12.01%
	Model I	6.255	3.501	8.94%	8.004	4.062	10.54%	9.499	4.595	12.54%
	Model II	6.276	3.225	8.49%	**7.262**	3.479	**9.67%**	**8.665**	4.146	**11.83%**

Figure 5 below illustrates a comparison of the loss function descent process in Model II and STGCN training process. This figure is the training session using METR-LA dataset with a time slice of 60 minutes. There are 250 training epochs. The images for other datasets and time slices are similar. Therefore, to avoid repetition, only Figure 5 is used for subsequent analysis.

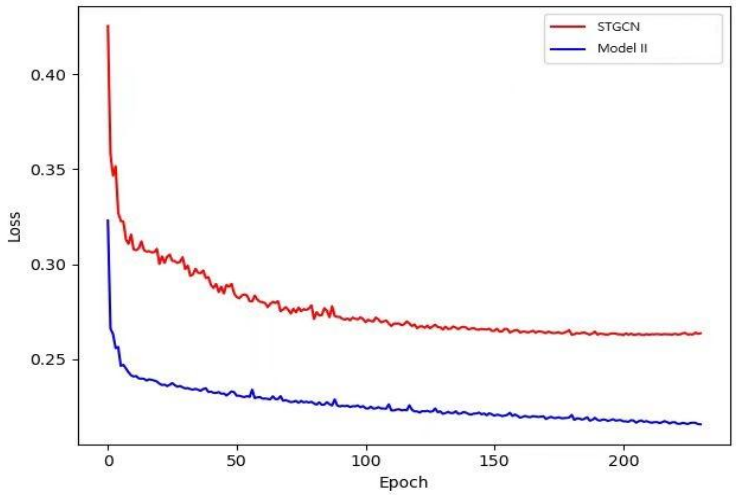

Figure 5: Loss function image for Model II and STGCN (METR-LA Dataset, 60mins)

As it can be seen in Table 2, Model II achieves the state-of-the-art results on six of the total nine test sets (three datasets * three time slices), with some of the results even outperforming baseline model by more than 10%. Model II also ranked in the top three on the remaining test sets. Combine with the loss function image shown in Figure 5, the effectiveness of Model II, or the k-SVD de-noising process, in traffic flow prediction tasks can be well illustrated.

Meanwhile, it can be seen from Table 2 that Model I is about 5% better than the STGCN model in most cases. Since Model I is not affected by the temporal module,

the necessity of considering multi-hop message passing in the traffic flow prediction task can be demonstrated; as well as the effectiveness of applying the k-SVD process as the ideal low-pass filter instead of p-hops message passing.

5.3 Analysis

This subsection further analyze the function of the model proposed in this study in traffic flow prediction through a few simple experiments. Figure 6 below shows the prediction results of target vertices from Model II on nine different datasets with different time slices compared to the actual traffic flow. The total duration is 1 day, which is 12*24 = 288 time slices.

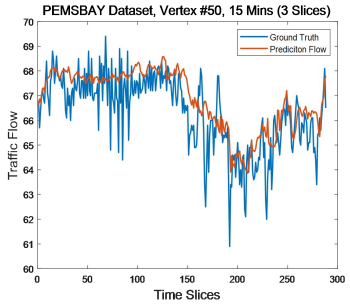

Figure 6(a)

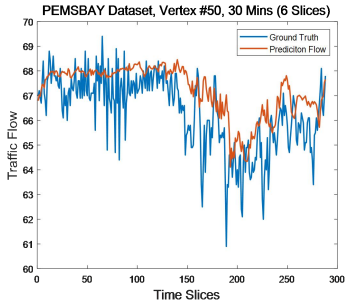

Figure 6(b)

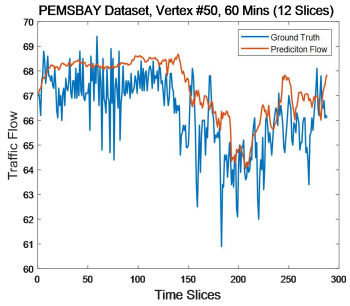

Figure 6(c)

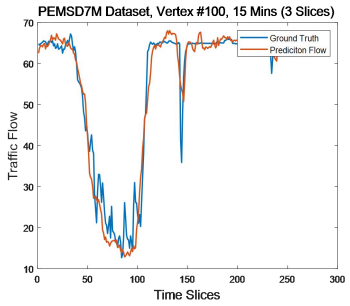

Figure 6(d)

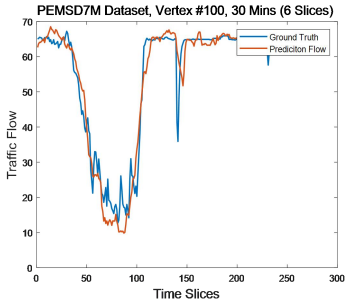

Figure 6(e)

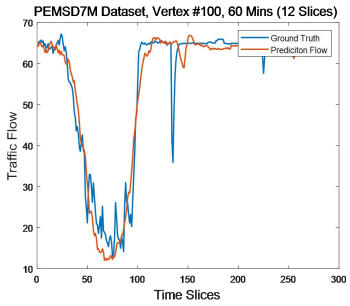

Figure 6(f)

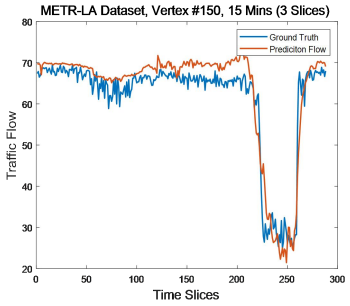

Figure 6(g)

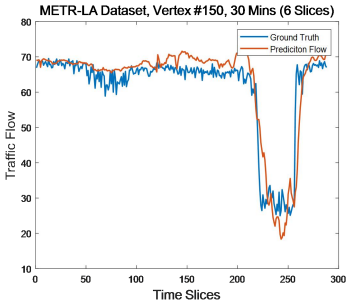

Figure 6(h)

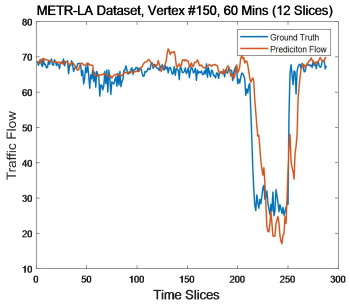

Figure 6(i)

Figure 6: Prediction results for Model II Vs. Ground truth

Figure 7-9 then compared the prediction results of different models in the same dataset, at the same time slice for all vertices. For clarity, only the 3 top ranked model are presented. The ground truth is distributed on the line of y = x. That is, the more convergent the results to the line y = x, the better the prediction performance is, and

vice versa.

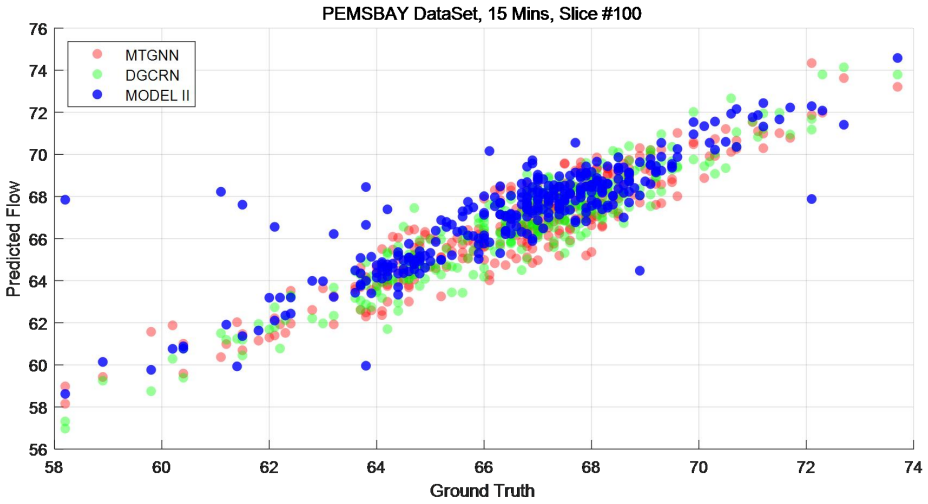

Figure 7: Predict result comparison on PEMSBAY dataset, Slice #100

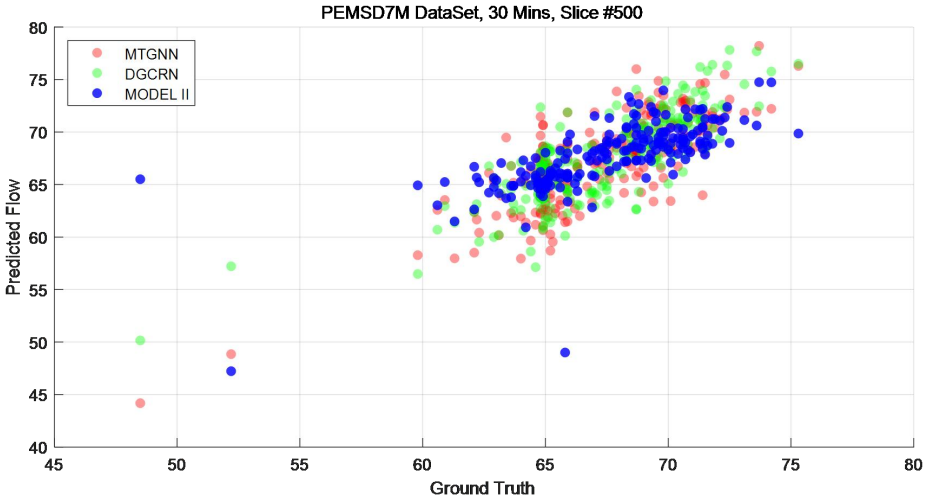

Figure 8: Predict result comparison on PEMSD7M dataset, Slice #500

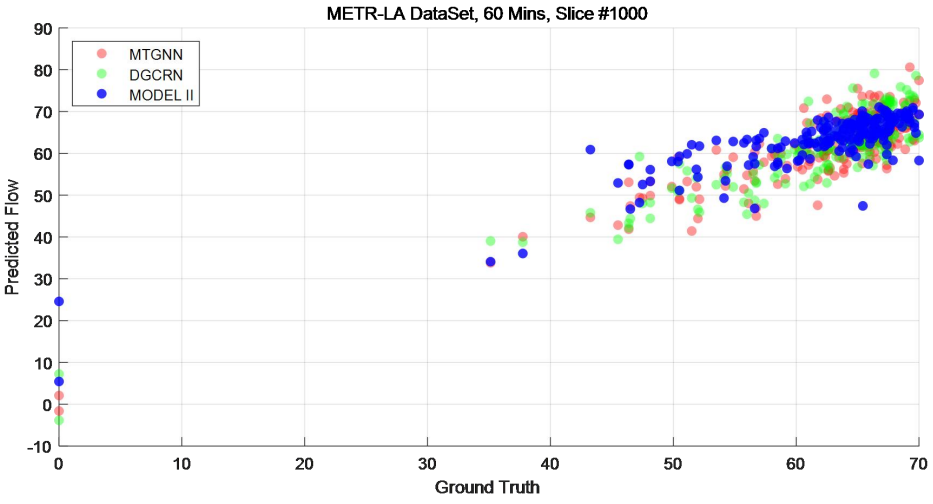

Figure 9: Predict result comparison on METR-LA dataset, Slice #1000

It can be seen from Figure 6 that from a time series perspective, Model II is able to extract the features and make reasonable prediction regardless of whether the traffic flow is in the peak, valley or normal range. However, the longer the time span of the prediction (from 3 slices to 6 slices then to 12 slices), the smoother the prediction results then to be. At this point, the model becomes less sensitive to small fluctuations and its predictions become worse. This is due to a shortcoming of the time module itself. The long-range dependence problem of time series analysis has been difficult to be solve effectively.

Moreover, it can be seen from Figure 7-9 that compared with the baseline models, the advantage of Model II is that the prediction results are more accurate when the flow of the target vertex is high. While at low flow vertex, Model II has a larger prediction error. This may be due to the fact that the basic concept of Model II is low-pass filtering. That is, the graph signal becomes smoother after pass through Model II. In spatial domain, it means that the predicted values tend to be similar at neighbor vertices. While in urban transport systems, nodes with high traffic flow tend to be concentrated in one or more hotspot areas and connected to each other. The flow rates of their nodes are also similar to each other. On the other hand, low flow vertices and non-hotspot areas tend to be dispersed around the hotspot areas, and are connected to each other by only a few nodes. This may be the reason why Model II has the characteristics reflected by Figure 7-9. In addition, another factor that affecting the performance of Modell II is that some of the predictions are much worse or even become outliers. If this problem can be solved effectively in further research, it may be possible to make the whole model perform better in traffic flow prediction tasks.

6. Conclusion

To summarize, by analyzing the patterns in related works, this study concludes that consideration of the message passing between multi-hop neighbors may have a more significant role in the traffic flow prediction tasks than capturing the message passing patterns between single-hop nodes more accurately. Then, based on spectral graph theory and graph signal processing, this study reveals that the principle of message passing between multi-hop neighbors is the low-pass filtering of graph signals, and proposes a method to fuse an ideal low-pass filter with a GCN model. Specifically, this model proposes a method which add a k-SVD de-nosing process after each training epoch of the STGCN model. Experiments prove that the addition of the de-noising process allows the STGCN model, which only has a simple spatial-temporal information extraction module, to perform close or even better than the existing state-of-the-art baseline models in traffic flow prediction tasks. Therefore, it is believed that this approach is an elegant and effective alternative to stacking complex deep leaning modules.

Meanwhile, the combination of de-noising process and deep learning model is a

promising idea. This study only applied a simple de-nosing process to an earlier existed model. As a result, the model has shortcomings resulting in less stable predictions. Even then, the performance of the model is competitive. It is believed that further process can be made in traffic flow prediction and other related tasks if it is based on advanced de-noising techniques or suitable modelling frameworks. At that time, instead of stacking deep leaning modules, maybe, De-noising Is All You Need!

$$\tilde{\mathrm{A}}\boldsymbol{x} = \begin{bmatrix} 1 & a_{12} & \cdots & a_{1n} \\ a_{21} & 1 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 + \sum_{i=1}^{n} a_{1i}x_i \\ x_2 + \sum_{i=1}^{n} a_{2i}x_i \\ \vdots \\ x_n + \sum_{i=1}^{n} a_{ni}x_i \end{bmatrix}$$

$$\left(\tilde{\mathrm{A}}\boldsymbol{x}\right)^T\left(\tilde{\mathrm{A}}\boldsymbol{x}\right) = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (x_i + \sum_{j=1}^{n} a_{1j}x_j)^{\mathbf{2}}$$

$$\mathbf{x}^T\left(\tilde{\mathrm{A}}\boldsymbol{x}\right)^T\left(\tilde{\mathrm{A}}\boldsymbol{x}\right)\boldsymbol{x} = \sum_{k=1}^{i} x_k^2 \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (x_i + \sum_{j=1}^{n} a_{1j}x_j)^{\mathbf{2}}$$

$$[\left(\tilde{\mathrm{A}}\boldsymbol{x}\right)^T\left(\tilde{\mathrm{A}}\boldsymbol{x}\right)]\boldsymbol{x} = \sigma^2\boldsymbol{x}$$

$$\sigma = \sqrt{1 + 2\sum_{i,j=1}^{n} a_{ij}x_ix_j + (\sum_{i,j=1}^{n} a_{ij}x_j)^2}$$