Investigation of Stability Property of Graph Neural Network Architectures under Domain Perturbations

Khoa Nguyen[1]

Supervisor(s): Dr. Elvin Isufi[1]**, Mohammad Sabbaqi**[2]**, Maosheng Yang**[3]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 24, 2024

Name of the student: Khoa Nguyen
Final project course: CSE3000 Research Project
Thesis committee: Dr. Elvin Isufi, Mohammad Sabbaqi, Maosheng Yang

Abstract

Graph Neural Network holds significant importance in various applications. Pioneering research has demonstrated state-of-the-art performance in practical applications such as Fraud Detection, Recommender Systems, or Traffic Forecasting by utilizing various Graph Neural Networks (GNNs) architectures. For these applications, one of the most important properties that needs to hold is the stability of GNN under stochastic perturbation as real-life networks undergo changes in topology on a frequent basis. However, it remains unclear how different architectures preserve this property under different perturbations. In this research, we aim to shed light on if this stability property undergoes drastic changes in the graph underlying topology, and if it affects the overall performance of the GNN in Traffic Forecasting problems. We demonstrate that the architectures differ in the stability property measured by different metrics, while some architectures retains their state-of-the-art performance, providing useful insight on the analysis of stability property on different graph neural network architectures in Traffic Forecasting problem.

1 Introduction

Background. Graph neural networks (GNNs) have become the state-of-the-art tool for extracting meaningful patterns from network-based data. Network-based data is pervasive and exists in many shapes and forms, arising from simple to intricate structures, prevalent in social networks, brain networks, finance, and more. They excel in tasks such as node classification, link prediction, and graph classification, driving advancements in major tech companies like Twitter (now X) for fake news detection [10], Pinterest for pin recommendations [21], while researchers are picking up pace on urban planning for complicated traffic forecasting.

Current GNN models assume a perfectly known network structure, overlooking practical issues like estimation errors, adversarial attacks, or network changes. For example in traffic analysis, road structures can undergo discrepancies like traffic accidents, road maintenance, and re-routing plans, effectively means changes in the network topology. This shift between training and testing networks leads to performance degradation for real-world applications of traffic forecasting, raising the importance of GNN stability analysis under domain perturbations.

In recent years, the interest in going deeper into the research field of graph stability property has been on the rise, work done by [8] researched the stability of Graph Convolutional Networks (GCN) to stochastic perturbations, [12] on GCN to edge rewiring and [6] on the general stability properties of GCN with Lipschitz filter.

Prior studies have focused on specific perturbations of a specific architecture, or specific types of perturbation such as edge rewiring. However, despite these advancements, a fundamental question remains unanswered: How stable do different GNN architectures perform in the face of stochastic perturbation?

Research questions and main contributions. This research aims to deepen the understanding of GNN stability in stochastic perturbations for various of architectures for perturbation techniques, mimicking the practical events of topology shifts in road networks. We will explore various GNN models and their performance practically for the node regression tasks, as it is one the most used tasks to capture the relationship of spatio-temporal in the network, accounting for uncertainties in the network structure under topology shift.

Roadmap. This paper includes the following sections. Section 2 provides an overview of relevant literature. Section 3 explains the research approach for analyzing the stability properties of different graph architectures. Section 4 presents the conducted experiments and their results. Section 6 discusses and interprets the findings. Section 5 addresses the study's ethical implications. Finally, Section 7 summarizes the results, limitations, and future improvements.

2 Related Work

This section provides an overview of relevant research on the stability property of graph neural network architectures under perturbation. The relevant researches can be divided into two sub-sections: controlled perturbation - which retains small and controlled perturbation rate, assuming the topology shift in the underlying graph embeddings have small changes. **Stochastic perturbation - our focus**, with random mode of changing the graph topology from small to large perturbation.

2.1 Controlled Perturbation

In recent times, there has been a focus on analyzing the stability of Graph Convolutional Neural Networks (GCNNs) to topological perturbations. Graph wavelet filters [9] were used in [25] to investigate the durability of graph scattering transformations in non-trainable graph neural networks in response to changes in the graph topology. Expanding on this, [7] examined stability to graph deformations measured by diffusion distance [2] and concentrated on diffusion wavelets [3]. By switching to trainable designs, [14] showed that GCNNs keep their stability in the face of topological perturbations, producing consistent representations for graphs that describe the same thing. In contrast, [6], [18] investigated the stability of GCNNs under relative perturbations and discovered that GCNNs can be efficient in discriminating high-frequency graph information while also being robust to perturbations. Building on these discoveries, [19] investigated stability in the graphon neural network, in which the graphon is the limit of graph sequences that converge. Moreover, studies have examined how resilient GCNNs are against focused attacks on the underlying graph. While [4] used reinforcement learning to construct attacks on both node-level and graph-level classification tasks, [26] focused on creating adversarial attacks on graph edges and node signals to misclassify target node labels. Simultaneously, [1] investigated

the resilience of GCNN against attacks on a subset of graph edges, proving that node labels do not change in node classification tasks.

2.2 Stochastic Perturbation

Advancing from the well-studied controlled perturbation, [8] investigated the GCNN architecture and its GCNConv filter under stochastic perturbation to discover the stability property is disproportionate with the complexity of constant in Lipschitz filter, and with GCNN having more layers and filters in its hidden state. While [13] investigated stability of GCNN under stochastic perturbation with random input graphs to conclude that GNN efficiently and reliably performed under random large input graph. Both of these perturbations method however only analyze the stability of a particular subset of filters with mathematical methods.

3 Methodology

In this section, we propose the following experiment procedure as shown in Figure 1 to investigate the stability property of GNN architectures. It aims to thoroughly identify and investigate how important the architecture is to graph topology perturbation, as well as to pick out the best-performing architecture (if any) and to raise hypotheses to them. The experimental methodology is divided into several sub-sections, each addressing a specific stage of the research process. Subsection 3.1 provides comprehensive details regarding investigating practical problems to solve. This practical problem will serve as the foundation to decide the subset of architectures we would investigate on. Subsection 3.2 focuses on deciding the subset of architectures under investigation, relying on the aforementioned defined practical problem. This section is a significant decider as we would avoid assessing under-performing models in this practical problem. Subsection 3.2 will conduct evaluation metrics for each of the models, its specifications, and statistical details about the underlying structure. This assessment aims to quantify the performance of each architecture, a stepping stone for the next sub-section, 3.3. Subsection 3.3 will investigate methods to compile and conclude the performance between architectures, furthermore analyze, explain, and compare the performance of each architecture among themselves.

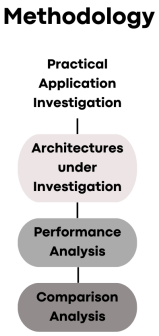

Figure 1: The research methodology employed to investigate the stability property between different GNN architectures.

3.1 Practical Application Investigation

Graph neural networks can solve a variety of practical problems, such as fraud detection on platforms like Twitter (or X), recommendation systems, and image processing. However, most of these applications are already well-researched. Traffic prediction, on the other hand, remains a challenging problem because it typically involves a two-step model (Figure 2) that first captures spatial information and then passes it to another model to capture temporal information, and this relationship demonstrated great complexity. According to a comprehensive study done by [11], the notable problem with Traffic forecasting arises in the spatial captured domain of GNN. Although numerous solutions have been suggested to address the issue of time dependency, such as recurrent neural networks and temporal convolutional networks, the challenge of capturing and modeling spatial dependency remains unresolved as spatial dependency involves the complex and nonlinear relationship between the traffic state at a specific location and those at other locations, displaying the heterogeneous nature of this problem.
Examples of such incidents include road construction, accidents, or the addition of new roads to the road-wide network. These changes can perfectly be mapped to link deletion, node deletion, or link addition in topology perturbation, furthermore supporting the understanding of the stability property of GNN under perturbation in the traffic prediction field, and of GNN architectures as a whole in the future.
For this reason, this research will focus on investigating different architectures under domain perturbation in the **spatial** capturing layer of different spatial-temporal GNN.

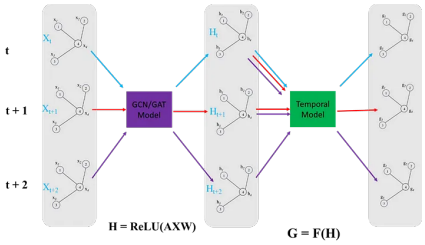

Figure 2: Spatial (purple) - Temporal (green) Graph Neural Network

3.2 Architectures under Investigation

Overall, there are 4 notable mechanisms (Convolution layers) for capturing spatial relationships in a road network according to [17], and their respective architecture: DiffConv [1] in DCRNN architecture [15], GCNConv [2] in A3TGCN architecture [24], and GATConv [3] in GMAN architecture [23]. Furthermore, with the promising TAGConv [4] from [5] which adapts to the topology of the graph when they scan the graph for computation, this TAGConv layer will be included in A3TGCN to replace its GCNConv layer since TAGConv was developed based on GCNConv, with different K (K = 1 for

[1] Diffusion Convolution Layer
[2] Graph Convolution Network Convolution Layer
[3] Graph Attention Network Convolution Layer
[4] Topology Adaptive Graph Convolution Layer

GCNConv and K ≥ 2 for TAGConv). However, since traffic dataset is often massive, requiring a large training time with dedicated equipment (such as a high-performance GPU), for this, we concluded the **3 architectures under investigation:**

1. A3TGCN
2. A3TTagConv
3. GMAN

Since three of these are state-of-the-art architectures with good running time, in additionally, the implementation of these three models are straight forward as per [17]. We will address the investigation of more architectures under discussion to improve the result of this work.

3.3 Performance Analysis

We only consider to capture the spatial nature of the architecture, hence we will implement a hook to capture the result after forwarding the input through the convolution layers (this will be called graph embedding from hereon) after data passes through the spatial layer. The algorithm for the perturbation of the input can be described by the following pseudo-code:

Algorithm 1 Perturb Adjacency Matrix by Percentage

Require: the adjacency matrix, adding edge percentage, removing edge percentage

number_of_edges $\leftarrow$ (count from the adj_matrix)
num_add $\leftarrow$ int(current_edges $\times$ add_percentage)
num_remove $\leftarrow$ int(current_edges $\times$ remove_percentage)

add_edges(adj_matrix, num_add)
remove_edges(adj_matrix, num_remove)

The retained result after passing the input through the feed-forward of the spatial layer will be a tensor in the following format, corresponding to the size of the output argument in the convolution layer:

$$\text{Shape(X)} = [N, F_{output}]$$

where N stands for number of nodes, and F stands for number of output layer in the convolution layer.

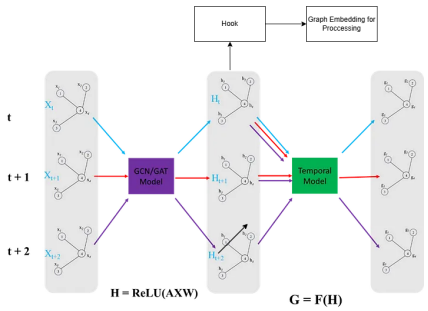

Figure 3: Hook to capture graph embedding

Overall, we represent A as graph embedding before the input graph being perturbed, while B is the graph embedding of the perturbed input graph (further discussion and algorithm can be found in 4. With this, we have two options to compute the similarity score between embeddings:

1. $$\text{Cosine Similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

   Where: - $\mathbf{A} \cdot \mathbf{B}$ is the dot product of vectors (graph embedding) $\mathbf{A}$ and $\mathbf{B}$. - $\|\mathbf{A}\|$ is the magnitude (or norm) of vector $\mathbf{A}$. - $\|\mathbf{B}\|$ is the magnitude (or norm) of vector $\mathbf{B}$. The result will range from -1 to 1, with 1 means two embeddings are identical ($\theta = 0$), 0 means no similarity and -1 indicates exact opposite between two embeddings.

2. $$d(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$$

   Where: - $\mathbf{A} = (A_1, A_2, \ldots, A_n)$ is the first point. - $\mathbf{B} = (B_1, B_2, \ldots, B_n)$ is the second point. - A_i and B_i are the coordinates of $\mathbf{A}$ and $\mathbf{B}$ respectively. The result will not be normalized as we are also interested in the difference of scales between different architectures. The larger the result, the greater the difference between two embeddings.

Each metric provides distinct information about the graphs. Cosine similarity captures the angular similarity between vectors, Euclidean distance measures the positional difference in the embedding space. Each metric has its own advantages and disadvantages. Therefore, we will include both to comprehensively assess the stability properties of the graph embedding before and after perturbation.

After capturing the stability property of each architecture, the data then passes through the pipeline to compute the evaluation **RMSE score** as traffic forecasting problem has learning task as **node regression**

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Where: - n is the number of observations. - y_i represents the actual values. - $\hat{y}_i$ represents the predicted values.

3.4 Comparison Analysis

The end result will be compared with the achieved RMSE score, and the increased percentage of the perturbation ratio to baseline models to determine which model can perform better under stochastic perturbation.

4 Experiments

In the following section, we display a series of experiments based on the methodology outlined in Chapter 3. Our aim is to describe the experiment process further to estimate the performance of all architectures under investigation. The following diagram shows the visualisation for the pipeline used in this experiment.

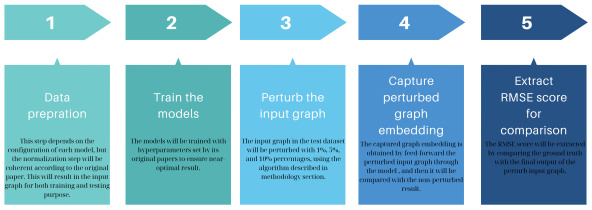

Figure 4: Visualisation of the experiment pipeline

4.1 Dataset

We conduct experiments on the real-world large-scale datasets collected by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS) called **PEMS-BAY**. 325 sensors were selected in the Bay Area and collected 6 months of data ranging from Jan 1st 2017 to May 31th 2017 for the experiment.

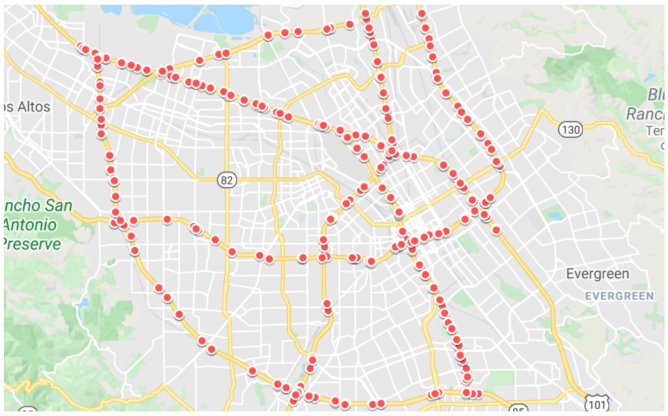

Figure 5: PEMS-BAY visualization [16] of 325 sensors distributed across California

In this dataset, the data preparation varies for each model due to the different data shapes required by each architecture. Despite these variations, we consistently employ the same methodology for sanitizing and normalizing the data, as outlined by [15], which serves as a standard procedure across all models. The total number of observed traffic data points is 16,937,179.

The dataset is aggregated traffic speed readings into 5 minutes windows, and apply Z-Score normalization. 70% of data is used for training, 20% are used for testing while the remaining 10% for validation. To construct the sensor graph, we compute the pairwise road network distances between sensors and build the adjacency matrix using thresholded Gaussian kernel.

$$W_{ij} = \begin{cases} \exp\left(-\frac{\text{dist}(v_i, v_j)^2}{\sigma^2}\right) & \text{if dist}(v_i, v_j) \leq \kappa, \\ 0 & \text{otherwise} \end{cases}$$

where W_{ij} represents the edge weight between sensor v_i and sensor v_j, $\text{dist}(v_i, v_j)$ denotes the road network distance from sensor v_i to sensor v_j. σ is the standard deviation of distances and κ is the threshold.

4.2 Experimental Setup

Each of the architecture required a separate way for preparing the input dimension for the learnable components, as well as different ways to interpret the stability score. Consequently, we will present the results in various formats (e.g., graphs, box plots) with different shapes and sizes, however, with the end goal of ensuring that the results are comprehensible.

A3TGCN and A3TTAG

The architecture has a spatial capturing mechanism at every time step X_t through a T-GCN model [22] that first capture both the spatial and temporal information of the data, then leverage the use of attention mechanism to detect pattern in the concatenated temporal and spatial space. Within the T-GCN model, we can place a hook at the GCN layer which handles the spatial capturing information.

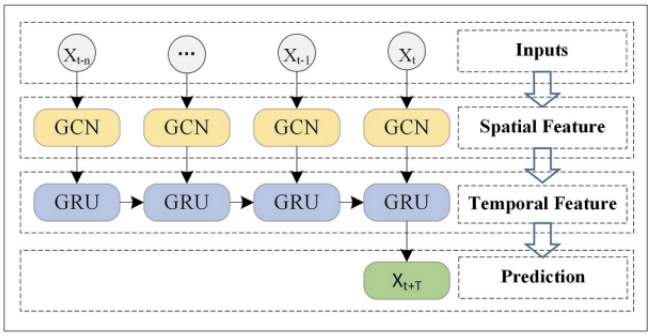

Figure 6: TGCN architecture [22], we will concentrate on the GCN layer, which is responsible for capturing spatial information.

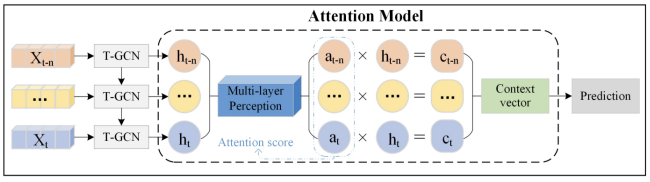

Figure 7: A3T-GCN architecture [24] visualisation, with concatenation of TGCN layers before passing through the attention model

The difference between A3TGCN and A3TTAG is in the spatial feature capturing mechanism, as we will replace GCN with TAGNET to evaluate the difference between these two architectures.

The data is prepared using the pytorch geometric temporal library, with the shape of the input described as

$$X \in R^{45000 \times 325 \times 212}$$

, and the output (prediction or ground truth) as

$$Y_{\text{prediction or ground truth}} \in R^{45000 \times 325 \times 12}$$

GMAN

Graph Multi-Attention Network [23] leverages the use of attention mechanism in GATConv in both spatial and temporal capturing domain. It follows a encoder-decoder structure, which consists of L ($L \in N^*$) Spatial-Temporal Attention Block (STAtt). Each of this STAtt block contains gated fusion of spatial and temporal attention layers. A transform attention layer is implemented between the encoder and decoder to convert the encoded traffic features for the decoder. Spatio-temporal embedding (STE) is used to integrate the input graph and time information into multi-attention layers. More detail can be found below with the inlustration of the architecture.

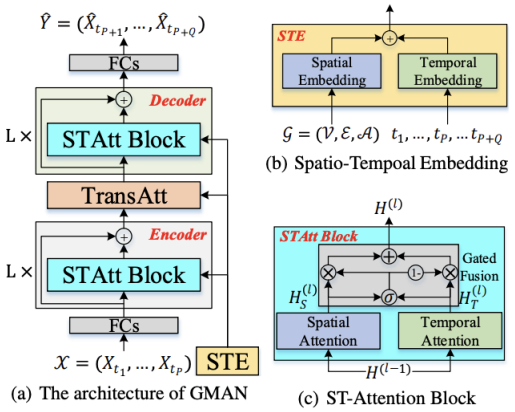

Figure 8: We will focus on the spatial capturing mechanism of the model [23], which happens in a STAtt block. A hook will be implemented to extract the spatial attention described here, in both encoder and decoder. The result will be averaged over 12 time steps and visualized.

GMAN is different than A3TGCN and A3TTAG especially in the data prepration step, as the architecture requires to perform *node2vec* approach to learn the vertex representations. In additionally, to simultaneously train the pre-learned vectors (obtained after applying node2vec) with the whole model, they will be fed into two-layer fully-connected neural network to concatenate the initial spatial and temporal embedding. Only then we obtain the STE input for the architecture, the input can be represented as $e_{vi}^S \in R^D$. Hence, for the data prepration step, we will first perturb the input graph using the algorithm specified in 3, then perform the node2vec incorporating with the two-layer fully connected neural network to attain STE. The reason why we will not study the effect of node2vec result (the initial Spatial Embedding) is the perturbation property of this graph embedding technique is well studied in [20], hence our main interest will be spatial attention mechanism in the inner structure of GMAN architecture.

4.3 Result

Peformance Analysis

Both A3TGCN and A3TTAG architectures show a high persistent to stochastic perturbation, while it is not the case for GMAN. The outcomes will be demonstrated by applying perturbations of 1% and 10%, as we can demonstrate meaningful observation with only two perturbation ratios as shown below. The similarity scores are depicted using both a histogram and a box plot, illustrating the overall distribution in the histogram and showing the similarity score for each of the 12 time steps in the boxplot.

A3TGCN

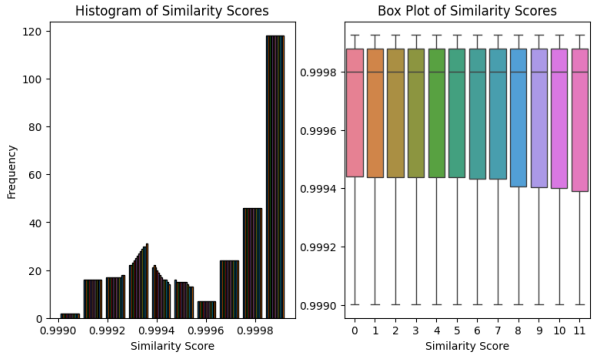

Figure 9: Perturbation score of **cosine similarity** with 1% removing and adding edges

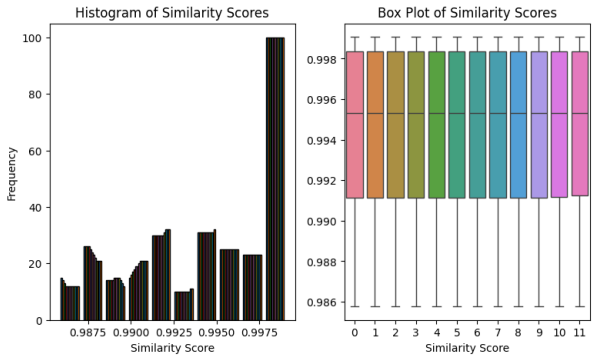

Figure 10: Perturbation score of **cosine similarity** with 10% removing and adding edges

As it can be seen from several perturbation iterations, the graph embedding similarity scores of cosine similarity are consistent, with the fluctuations are shown to be minor, ranging from 0.98 to 1 (highly stable)

This conclusion also comes with the euclidean distance:

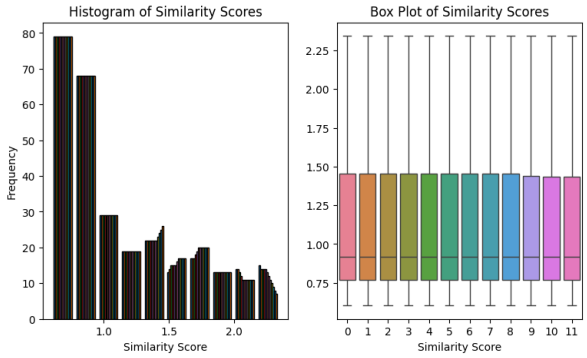

Figure 11: Perturbation score of **euclidean distance** with 1% removing and adding edges

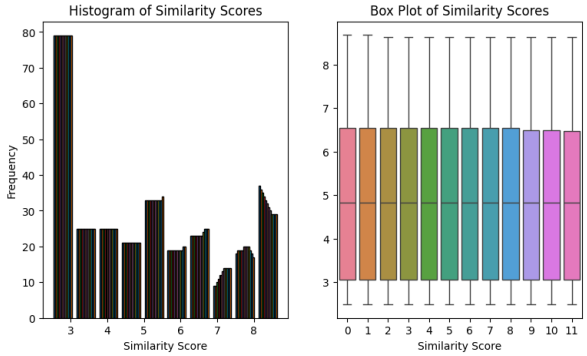

Figure 12: Perturbation score of **euclidean distance** with 10% removing and adding edges

The similarity score of eucledian distance following with cosine similarity fluctuates only around 0.75 and 1.5, displaying a **high** stability property in both metric scores.

A3TTAG

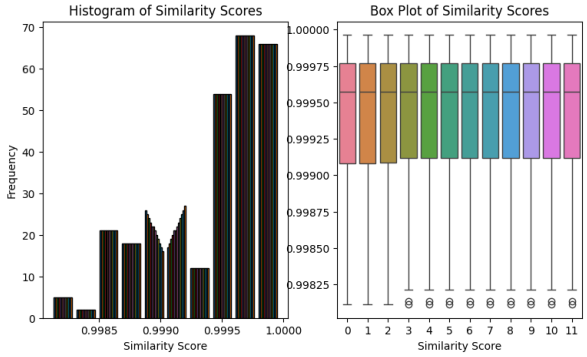

Figure 13: Perturbation score of **cosine similarity** with 1% removing and adding edges

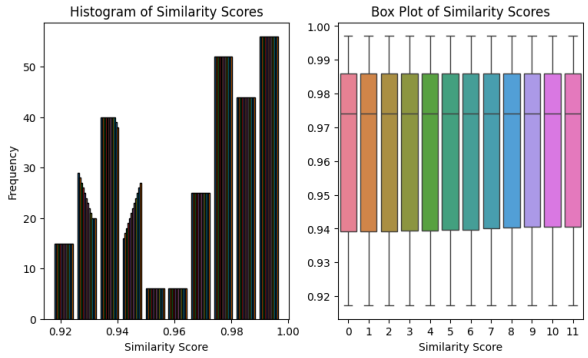

Figure 14: Perturbation score of **cosine similarity** with 10% removing and adding edges

The graph embedding similarity scores of cosine similarity are pertained, with the fluctuations are shown to be minor, ranging from 0.994 to 1 (highly stable). This conclusion however, differs with the euclidean distance, unlike to A3TGCN:

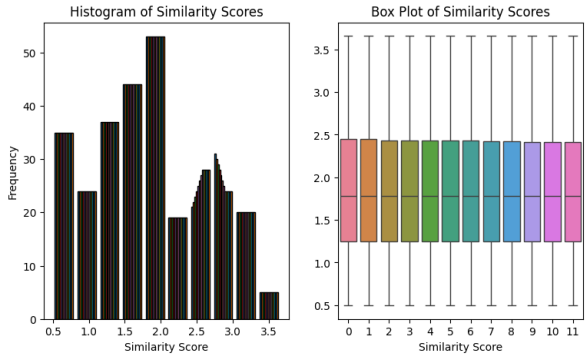

Figure 15: Perturbation score of **euclidean distance** with 1% removing and adding edges

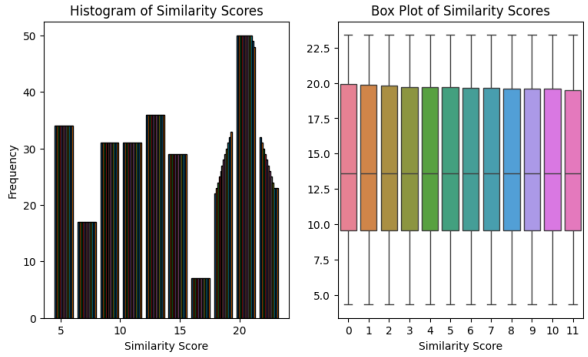

Figure 16: Perturbation score of **euclidean distance** with 10% removing and adding edges

The similarity score of euclidean distance differs significantly for A3TTAG, which fluctuates around the region of 5 and 23, following with cosine similarity fluctuates only around 0.75 and 1.5, displaying a high stability property in

only cosine similarity metric scores. The result shows a disparity in the stable property of A3TTAG, and can be interpreted as the domain capturing of A3TTAG is different in how the scale (as shown in the eucledian score above) of graph embedding passing through the spatial feature.

GMAN

GMAN, compares to the first two architectures, displays a high unstable property in both metrics

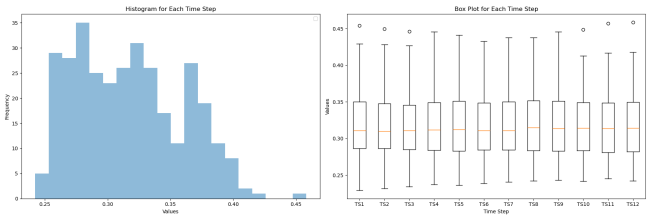

Figure 17: Perturbation score of **cosine similarity** with 1% removing and adding edges

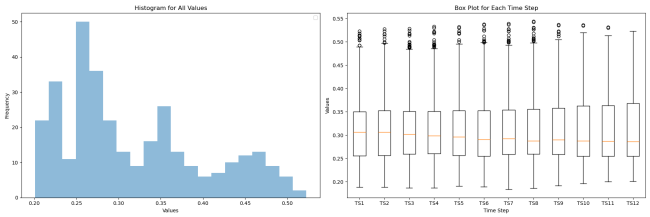

Figure 18: Perturbation score of **cosine similarity** with 10% removing and adding edges

In the case of 1% perturbation, the result ranges between 0.25 and 0.45, while for 10% perturbation, it ranges from 0.25 to 0.35. This significant difference in cosine similarity scores highlights substantial disparities in the spatial embeddings between perturbed and non-perturbed graphs within the GMAN architecture.

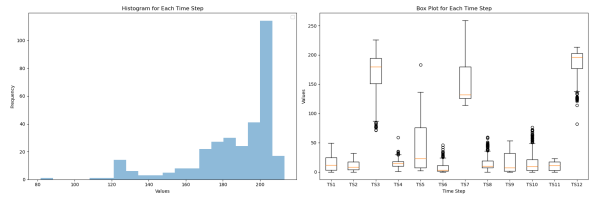

Figure 19: Perturbation score of **euclidean distance** with 1% removing and adding edges

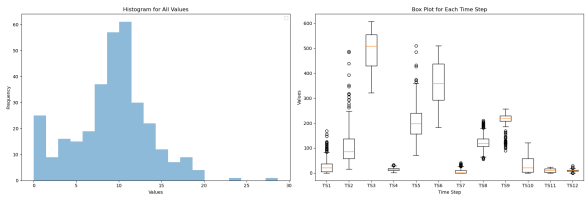

Figure 20: Perturbation score of **euclidean distance** with 10% removing and adding edges, the histogram result is truncated for the scale is above 100, hence the y column should be ranging from 100 to 600, correspondingly with the boxplot result.

For the Euclidean similarity score, the disparity increases notably, and the variance among the time steps also differ. With 1% perturbation, the scores vary from 50 to 250 across different time steps, indicating highly unstable scaling output. In contrast, with 10% perturbation, the scores range from 300 to 600, showing a significant increase compared to the 1% perturbation. These frequency ranges highlight the unstable nature of the spatial attention mechanism within the GMAN architecture

Comparison Analysis

To improve the illustration of the connection between the perturbation ratio and each model's final output performance, we add a 5% perturbation to the equation. The results show that A3TGCN and A3TTAG perform better than GMAN under stochastic edge perturbation. On the other hand, GMAN performs better at baseline than these two architectures. An illustration can be found in figure 21

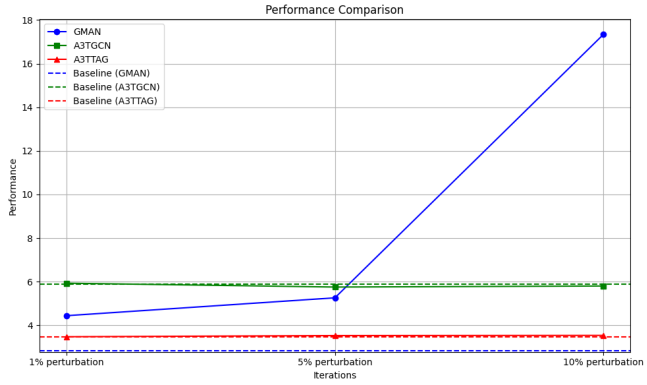

Figure 21: Performance comparison with RMSE scores between 3 architectures

Concluding from the Performance Analysis and RMSE score, we can show that for which architecture stable under which metric.

5 Responsible Research

This study utilizes traffic information through sensors that only capture speed and number of vehicles, without further information. Our findings are ensured to be reproducible with a functional personal computer (regardless of running time). In Section 5.1, we discuss the ethical considerations involved

Metric	A3TGCN	A3TTAG	GMAN
Cosine Similarity	Y	Y	N
Euclidean Distance	Y	N	N
RMSE score	Y	Y	N

Table 1: Interpretation and Comparison of the result achieved from analyzing three architectures. [Y] means Yes, indicating that the score is stable under 10% of max perturbation, while [N] means No, indicating that the score is unstable under 10% of max perturbation.

in the usage of traffic dataset such as PEMS-BAY. Additionally, Section 5.2 elaborates on the reproducibility of our research results and implementation. Furthermore, all models and codes have been cited, and credits have been given to the authors of the original works under Creative Copyrights.

5.1 Scientific Integrity

The PEMS-BAY dataset [15] only captures the speed and number of vehicles with 325 sensors, the data is anonymous hence poses no risk of leaking personal information, as the sensors did not capture datas in videos format nor personal vehicles' license plates. Furthermore, the data of the PEMS-BAY is a standard traffic forecasting dataset that contain no subjective metadata of traffic passing through 325 sensors, hence it is assumed that there exists no bias in the dataset.

5.2 Reproducibility

The methodology and algorithm in Section 3 are descriptive for the approach that we derived for the results, and, we publicly open-source the code on Github. In additionally, the architecture of A3TGCN and A3TTAG are implemented with the standard pytorch geometric temporal library, which is well documented and intuitive to understand and follow, more information can be found at [17]. The results have been achieved by running the experiments 10 times to detect if there is any significant change.

6 Discussion

This chapter discusses the findings as well as author's interpretation of the result, furthermore made some hypothesis as intitial step for further research in this field.

6.1 Possible explanation for why A3TGCN and A3TTAG performed better than GMAN

This finding in figure 21 implies that the A3T-GCN/TAG models successfully preserve the spatial-temporal embeddings of node features by using multi-layer perception to produce context vectors, as shown in Figure 7. As a result, when assessing new inputs, the vector data of the input is the only thing that is considered, making the graph's topology information unimportant for evaluation after training. To compute the final result, however, GMAN heavily depends on both spatial and temporal embeddings at each input step (see figure 8), indicating its dependency on node feature properties and graph topology. This argument can be made to research a multi-model that can perform well under both normal conditions and fault-prone environment, as the base score of GMAN is superior compared to A3TGCN and A3TTAG, but deteriorate significantly after a small percentages of stochastic perturbation.

6.2 Interpretation of the relationship between similarity metrics

The result from Table 1 however draws no correlation between spatial embedding between cosine similarity and euclidean distance to RMSE score. We argue that cosine similarity of the spatial embedding is a better metric to determine the final performance of the model under perturbation as both A3TGCN and A3TTAG display a superior performance compare to GMAN under stochastic perturbation. However, further research may be needed as more architectures should be investigated to conclude with a higher confidence level. This raises up a better investigation into multi-model traffic forecasting, as some model can perform significantly better under stochastic perturbation, and can be a mean to better increase the prediction of fault-prone data.s

7 Conclusions and Future Work

We have summarized the importance of stability research and examined the body of work that has already been done in this area. Our conversation emphasizes the necessity of creative methods in stability analysis to fill in the gaps and overcome the difficulties that the field is currently facing. We then looked into the stability of various architectures' spatial embeddings under stochastic edge perturbation for node regression, solving Traffic Forecasting problem through PEMS-BAY dataset. In conclusion, we presented a comparative analysis of each model's overall performance and made preliminary findings regarding the connection between spatial graph embeddings and overall graph performance following the stochastic perturbation. Although there is a lack of prior research on the relationship between spatial embedding and overall graph performance, our analysis has revealed significant differences in model performance, and display that some models are superior under stochastic perturbation. For future work, to fully comprehend the relationship between spatial embedding and overall graph performance, more models and settings should be investigated. Finally, future research could investigate possibilities of a multi-model traffic forecasting architecture, as real-data is fault-prone, if the responsible person detects there exists a failure in sensor, or streaming data which passing through experiences data corruption after a certain threshold, a more fault-tolerance model should be used instead.

References

[1] Aleksandar Bojchevski and Stephan Günnemann. Certifiable robustness to graph perturbations. *arXiv preprint arXiv:1910.14356*, 2019.

[2] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[3] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[4] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning (ICML)*, 2018.

[5] Jian Du, Shanghang Zhang, Guanhang Wu, Jose M. F. Moura, and Soummya Kar. Topology adaptive graph convolutional networks, 2018.

[6] Fabio Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

[7] Fabio Gama, Alejandro Ribeiro, and Joan Bruna. Diffusion scattering transforms on graphs. In *International Conference on Learning Representations (ICLR)*, 2019.

[8] Zhan Gao, Elvin Isufi, and Alejandro Ribeiro. Stability of graph convolutional neural networks to stochastic perturbations, 2021.

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

[10] Loukas Ilias and Ioanna Roussaki. Detecting malicious activity in twitter using deep learning techniques. *Applied Soft Computing*, 107:107360, 2021.

[11] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, November 2022.

[12] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. Interpretable stability bounds for spectral graph filters, 2021.

[13] Nicolas Keriven, Alberto Bietti, and Samuel Vaiter. Convergence and stability of graph convolutional networks on large random graphs, 2020.

[14] Ron Levie, Wenqiang Huang, Leonardo Bucci, Michael Bronstein, and Gitta Kutyniok. Transferability of spectral graph convolutional neural networks. *arXiv preprint arXiv:1907.12972*, 2019.

[15] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2018.

[16] Huakang Lu, Dongmin Huang, Youyi Song, Dazhi Jiang, Teng Zhou, and Jing Qin. St-trafficnet: A spatial-temporal deep learning network for traffic forecasting. *Electronics*, 9(9), 2020.

[17] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzmán López, Nicolas Collignon, and Rik Sarkar. Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models, 2021.

[18] Lucas Ruiz, Fabio Gama, and Alejandro Ribeiro. Graph neural networks: Architectures, stability and transferability. *arXiv preprint arXiv:2008.01767*, 2020.

[19] Lucas Ruiz, Zhengdao Wang, and Alejandro Ribeiro. Graph and graphon neural network stability. *arXiv preprint arXiv:2010.12529*, 2020.

[20] Zhi-Feng Wei, Pablo Moriano, and Ramakrishnan Kannan. Robustness of graph embedding methods for community detection, 2024.

[21] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems, 06 2018.

[22] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2020.

[23] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction, 2019.

[24] Jiawei Zhu, Yujiao Song, Ling Zhao, and Haifeng Li. A3t-gcn: Attention temporal graph convolutional network for traffic forecasting, 2020.

[25] Dan Zou and Gilad Lerman. Graph convolutional neural networks via scattering. *Applied and Computational Harmonic Analysis*, 49(3):1046–1074, 2020.

[26] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *International Conference on Knowledge Discovery and Data Mining (KDDM)*, 2018.