Stability of Graph Neural Network with respect to different types of topological perturbations

Alex Brown[1]

Supervisor(s): Elvin Isufi, Mohammad Sabbaqi, Moasheng Yang

[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 23, 2024

Name of the student: Alex Brown
Final project course: CSE3000 Research Project
Thesis committee: Elvin Isufi, Mohammad Sabbaqi, Moasheng Yang, Klaus Hildebrandt

Abstract

Graph Neural Networks are widely used as useful tools to investigate graphs because they can learn from the topological structure of graphs. In practical applications, the graph's structure can change over time, have errors or be subject to adversarial attacks. These perturbations negatively impact the accuracy of the neural network. The theoretical stability of graph neural networks has been analysed already and in this paper, the stability of graph neural networks is investigated experimentally. The performance of different perturbation strategies is compared to see how different perturbations impact stability.

1 Introduction

Graph data structures are widely used since they capture non-euclidean relational data. Graphs are the de facto standard in several fields like social networks, recommender systems, and molecule classification [1]. Researchers have developed graph neural networks (GNNs) that can operate on graphs and have been shown to have immense predictive power in a variety of learning tasks [2].

Conventional neural networks do not perform well on graphs. Neural networks assume a fixed ordering of, and a relation between, every input. To input a graph in the model it needs to be embedded in a feature vector. To convert a graph to a vector you can save the value at every node in an order vector. Any ordering of the nodes results in a valid embedding but the model will have a different output. In neural networks, layers are fully connected or connected with a convolutional filter. Both assume that a regular structure and that nodes are connected according to a pattern. Graphs do not have a regular structure, the edges connecting the nodes are the structure and conventional neural networks cannot model it. Additionally, graph-level tasks work on graphs with an arbitrary size input which regular neural networks cannot do [3].

To address these issues GNNs were developed. In this paper, Graph Convolutional Networks (GCNs) are considered. A GCN is a cascade of layers, where every layer is composed of a graph convolution with a polynomial filter and a pointwise non-linearity [1] [2]. Graph convolution with polynomial filters can be defined as $x' = x \sum_{i=0}^{d} w_i L^i$ where x is the input, L is the Laplacian representation of the graph, d is the degree of the filter and w are the filter coefficients [3]. Depending on the task the final layer can consist of various other types of layers including a pooling layer or a classifier. Variations have been made to this model that do not use convolution but similar functions that do maintain node-order equivariance [4] [5].

The experiments are performed with a Topology adaptive graph convolutional network (TAGConv network). TAGConv is a GNN model that aims to maximize the use of the irregular structure of graph data. The convolution is approximated by only considering the K^{th} nearest neighbours. The next layer of the network is calculated with:

$$X' = \sum_{k=0}^{K} (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})^k X W^{(k)} \tag{1}$$

where D is the degree matrix, A the adjacency matrix, X the graph signal and $W^{(k)}$). TAGConv reaches a higher accuracy compared to other methods that approximate the graph convolution with significantly less computing power.

Research into GNN design is still very relevant and popular in the machine learning community. Yet the stability of GNNs is still poorly understood. In practical applications, there will always be some inaccuracies in the graph structure. This can be caused by the data changing after the model has been trained, the graph can be based on a measurement with errors, or the graph can be manipulated by adversarial attacks. All of these cases result in a graph topology that is different from the one that the graph was trained on. These errors are called perturbations and negatively impact the accuracy of the model.

There have been key papers investigating the theoretical stability bounds of GNNs and graph filters. Kenlay et al. researched the stability bounds of linear graph filters and the tightness of these bounds in [6]. In [2] the theoretical bounds of GNNs are proven and the implications for GNN design are discussed. In the paper by Xu et al. a method is proposed to train a model in such a way that it is more robust to adversarial attacks [7]. Our contribution is to experimentally investigate the tightness of the stability bounds with respect to different types of perturbation strategies. In [6] the bound is based on the matrix norm of the perturbation. The goal is to compare the impact on the stability of different perturbation strategies and check whether there are special cases where perturbations have more impact on the stability than would be expected based on the matrix norm.

In section 2, we explain the approach and the theoretical basis of the experiments. Then the experimental set-up is fully described in section 3. The results of the experiments are shown in section 4. In section 5, the ethical aspects of the research are discussed. Finally, the impact of the results and recommendations for future research are discussed in section 6.

2 Approach

To test the stability, the performance of the unperturbed and perturbed graphs is compared on the Cora dataset [8]. This dataset is used for node classification and the dataset is split into a train, validation, and test set. Broadly, the approach to testing the stability is as follows. Firstly, the GNN is trained on the original dataset. Secondly, the dataset is perturbed with various strategies. Finally, the GNN predicts the test set of both the unperturbed and perturbed graphs.

Stability can not be measured directly and so we use metrics from which we can make conclusion about the stability. The main metrics used in this study to evaluate the performance are the accuracy, the relative euclidian distance of the unperturbed and perturbed graphs, and the relative distance normalised by the norm of the error matrix. The accuracy is defined as

$$acc = (\hat{y} - \bar{y})/len(y) \tag{2}$$

where $\hat{y}$ is the predicted classes and $\bar{y}$. The relative difference is given by

$$err = ||y - y_p||/||y|| \tag{3}$$

where y is the output of the GNN and y_p is the output of the GNN with the perturbed graph. The error matrix is defined as the absolute difference between the matrix representation of the two graphs

$$\mathbf{E} = |\mathbf{S} - \mathbf{S_p}| \tag{4}$$

where $\mathbf{S}$ is the matrix representation of the graph. For GNNs, the stability is bounded by:

$$||g(\mathbf{S}) - g(\mathbf{S_p})|| \leq \Delta L F^{L-1} \epsilon + \mathcal{O}(\epsilon^2) \tag{5}$$

where $g(\mathbf{S})$ is the output of the GNN on the data $\mathbf{S}$, ϵ is some constant for which $\epsilon \geq ||\mathbf{E}||$, L is the number of layers in the model, F is the number of features, and Δ is the stability constant of the graph. [2]. The stability is approximated experimentally by

$$C \geq ||g(\mathbf{S}) - g(\mathbf{S_p})||/||\mathbf{E}|| \tag{6}$$

with $C = \Delta L F^{L-1}$. By dividing the relative distance by the norm of the error matrix we can compare the impact of different strategies even if some have larger perturbations.

There are a limited number of perturbation operations that can be applied to graphs. Firstly, edge weight perturbation where the weight of an edge is changed. Secondly, adding or deleting an edge which technically is a special case of the edge weight perturbation where the weight is set to zero or non-zero. Finally, Adding or removing a node from the graph. In this paper, we focus on edge addition and removal.

These operations can be applied in many different ways to varying effects; the goal is to compare different strategies of perturbations. Now we define the strategies for applying these operations.

- Add Random, which randomly selects edges to add to the graph.
- Delete Random, which randomly removes edges.
- Add-Delete Random which uses half of its budget to add edges and half to remove edges.
- Rewire, which randomly deletes two edges and then connects the previously disconnected nodes with two new edges. This operation keeps the degree, the amount of edges the node is connected to, of all the nodes intact while changing the topology.
- Remove Node, which selects a node and removes all of the edges connected to it. This does keep the node in the graph and the accuracy of that node will be very low.
- Robust, which considers all possible additions and deletions in the graph and adds or deletes the edges which would result in the lowest spectral norm of the error matrix $||E||_2$. This should result in a very small impact on the stability.

For the Add and Delete random strategies, we have also investigated the degree of the nodes where edges are added or removed. The hypothesis is that adding or removing edges adjacent to nodes with a high degree has less impact on the stability as is the case in linear filters [6]. The contribution of one edge to the new embedding of the node is inversely proportional to the amount of connected edges. Nodes with a lower degree are thus more impacted by the addition or removal of an edge.

3 Methods

3.1 GNN model

The GNN architecture consists of three TAGConv layers, as implemented in Pytorch geometrics. The non-linearity used is the hyperbolic tangent. The layers have 32, 16, and 8 nodes respectively. The learning rate is 0.01, the weight decay is 0.0005. The model was trained using a cross-entropy loss function and with 25 epochs.

The experiment is done on the Cora dataset [8]. The dataset contains 2708 nodes and 5278 undirected edges. The model will perform node classification on this dataset. The data is stored as a PyTorch tensor with two lists. Every edge is represented as an integer in both lists. The first indicates which node the edge is coming from and the second is the index of the second node. Since the graph is undirected every edge is saved twice with one in the opposite direction.

For the first experiment, the graph is perturbed with an intensity of 20 and all the strategies are compared with each other. The effect of the degree on the stability is determined by plotting the degree of affected nodes and the relative distance. This is done for the Add Random and Delete Random strategies at an intensity of 1 and 100. Lastly, the effect of perturbation intensity is explored by analysing stability with varying perturbation intensities.

3.2 Perturbation algorithms

Add Random

The add random strategy selects two random numbers between zero and the number of nodes and checks if an edge exists between them. If not, it adds an edge and increases a counter. When this counter equals the budget assigned to the perturbation, the algorithm returns the new graph and terminates.

Delete Random

The delete random strategy selects a number between zero and the number of edges and then adds that edge to a mask. The algorithm then applies the mask to the edge index and returns the new graph without the selected edges.

Add Delete Random

Add delete random assigns half of its budget to adding edges randomly and half to deleting edges. This is done using the algorithms described above. None of the edges that were added by this algorithm can be removed again or vice versa.

Rewire

This strategy randomly selects two edges that do not share any nodes. These two edges are then deleted and two new edges are made that connect two previously disconnected nodes.

Remove Node

This strategy randomly selects a node and finds all of the edges that are connected to it. All of these edges are then removed from the edge index. This operation still leaves the selected node in the graph with no connected edges.

Robust

The robust strategy adds or deletes the edge that minimises the norm of the error matrix. Each iteration consists of choosing an edge to perturb, calculating the norm of the error matrix, comparing the norm with the previous minimum and saving the edge with a minimal norm. Ideally, this algorithm considers every edge in the graph and so is deterministic. We have used a deterministic version that considers x edges of which half are guaranteed to delete the edge to ensure that some edges considered are deletions since the graph is sparse.

3.3 Evaluation

After the graph has been perturbed, the model's performance can be assessed with multiple metrics. By inputting the unperturbed and perturbed graph into the model we get two lists of feature vectors detailing the probability that a certain node is a certain class. By comparing these two lists we can quantify the system's stability. Firstly, the accuracy is calculated by selecting the maximum value of each feature vector as the assigned class at that node and comparing it with the ground truth. The average score for the training, validation and test data is calculated.

Secondly, the relative distance of the output vectors is calculated. The second norm of the difference between the perturbed and unperturbed output is divided by the norm of the unperturbed data. This results in one scalar indicating the effect on the stability.

Lastly, dividing the norm of the difference of feature vectors by the matrix norm of the error matrix results in a lower bound for the stability bound of the system. The error matrix is obtained by calculating the difference between the Laplacian of the perturbed and unperturbed graphs.

4 Results

First, the experiments are performed with the same perturbation intensity for every strategy to compare the accuracy, relative distance and normalised relative distance. The results can be seen in fig: 14, 2, and 3

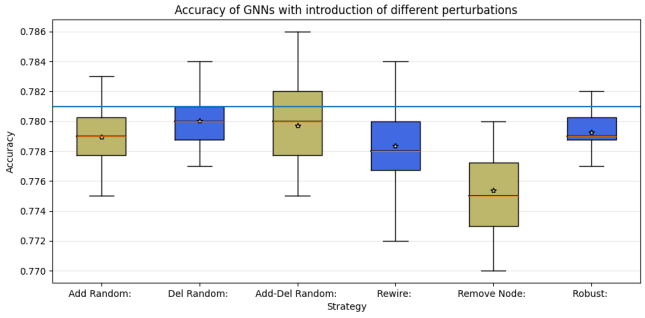

Figure 1: Accuracy of different strategies

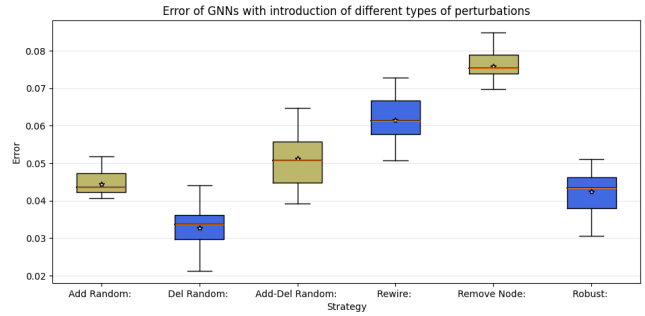

Figure 2: Relative distance of different strategies

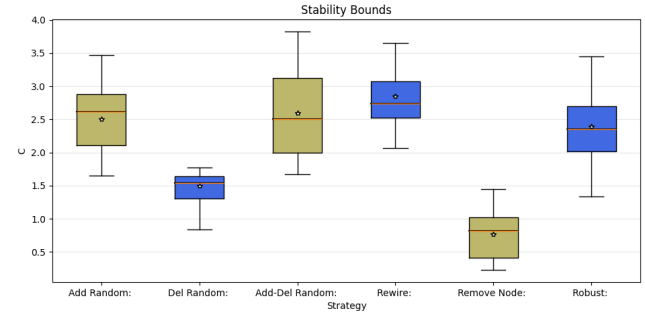

Figure 3: Stability bounds of the different strategies

Some of the perturbations can be seen to increase the stability. This is consistent with the literature where adding perturbations is used to improve the generalizability of models [9]. The improvement depends on the training of the model and the intensity of perturbations.

The accuracy and the relative distance have an inverse relation. The relative distance is a measure of how much

the perturbed output differs from the original. It does not indicate if the perturbed output is closer to or further away from the ground truth. There is a correlation between the mean relative distance and the accuracy. The error has less variation than the accuracy and there is a larger difference between the strategies.

Adding an edge is more detrimental to the stability than removing one. In this experiment, the edges are added randomly without domain knowledge. If the edge connects two completely unrelated nodes the model will extract information from the adjacent node. Deleting an edge means that there is less information to learn from but not any false information. Our conclusion is that this is not a general rule and to generalise the results, one needs to investigate more datasets and models. Depending on the domain and the nature of the perturbations the impact of an edge addition or removal can significantly vary.

The Add Delete Random and the Rewire strategies have the largest variation in both accuracy and relative distance. The budget for all operations is the same but per strategy, the size of the operations varies. In the Add Delete Random strategy, one operation adds and deletes one edge and the Rewire strategy adds and deletes two. Both strategies are special cases of the Add and Delete operations and the mean accuracy is similar.

From fig 3 it is clear that the Delete Random and the Remove Node strategies perform the best when normalised with the error norm of the matrix. The matrix norm of the error is very large for Remove Node. The other strategies perform similarly with an average C of 2.5 to 3.

For the second experiment, we compare the stability of different strategies over various perturbation intensities. Only the Add Random strategy is displayed in fig 4 and 5 and Delete Random can be found in the Appendix. All of the other strategies display similar behaviour. The stability is approximately linear with the perturbation intensity. This is in line with the theory since the stability bound is linear with the norm of the error [2]. The accuracy, relative distance, error norm and stability bounds all grow linearly with the intensity.

Finally, the relation between the degree of affected nodes and stability. For this experiment, we consider the Add Random and Delete Random strategies with an intensity of one edge. We only consider the primitive operations of Add Random and Delete Random here. In fig 6 and 7 the relative distance is plotted over the average degree of the affected nodes before perturbing.

When adding one edge there is a significant correlation between the relative distance and the degree of the nodes. Nodes with a lower degree have a much larger impact on the stability on average. But when multiple edges are perturbed this correlation disappears as can be seen in fig 8 and 9. Here 100 edges were perturbed at a time and the distribution seems much more like a normal distribution. The Delete Random strategy does have a lower error than the Add Random and a higher average degree. This is a possible explanation for Delete Random outperforming the Add Random strategy.

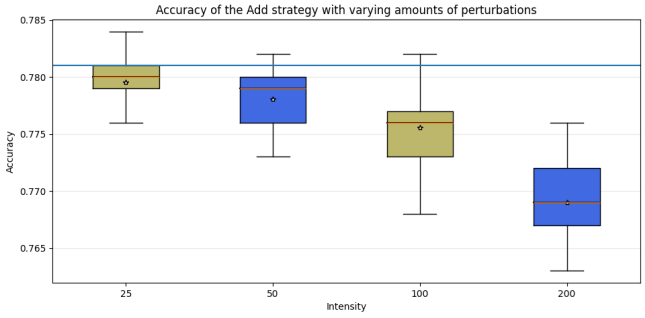

Figure 4: Accuracy over intensity with Add Random

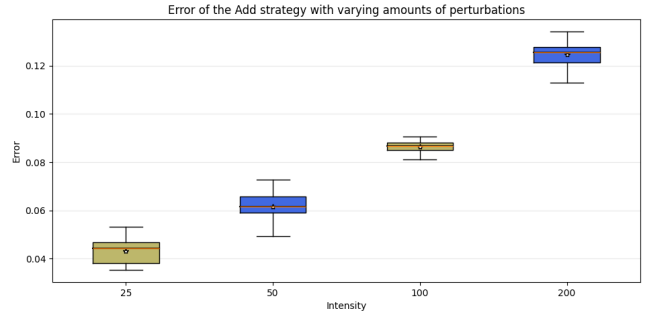

Figure 5: Relative distance over intensity with Add Random

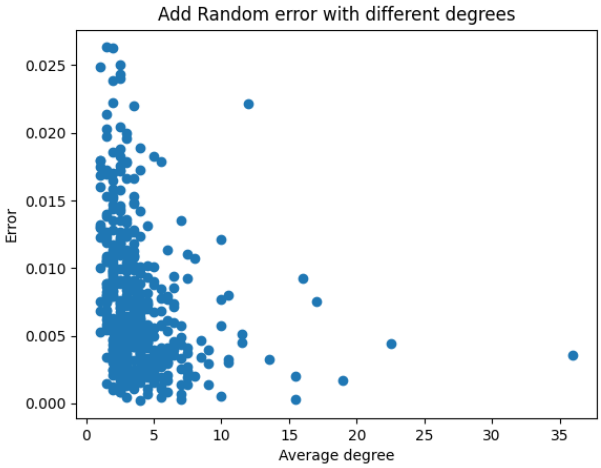

Figure 6: Relative distance of Add random with one edge addition

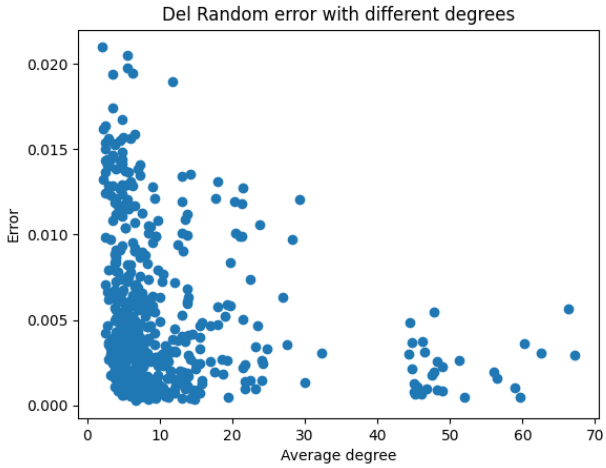

Figure 7: Relative distance of Delete random with one edge deletion

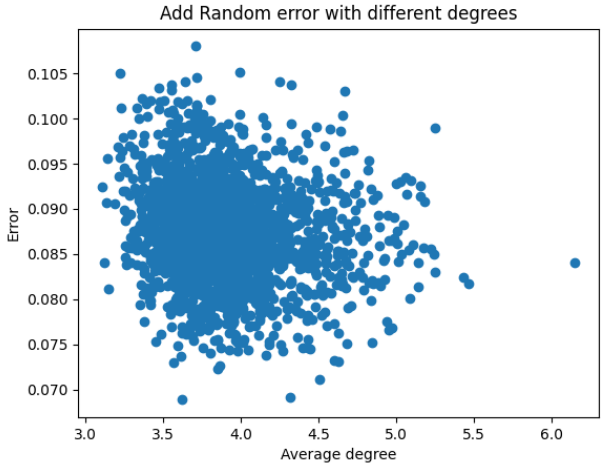

Figure 8: Relative distance of Add random with 100 edge additions

Since we only consider one edge removal the other strategies will perform the same but will just collect data with a different distribution. The results are shown in 6 and 7. There is a relation between the degree and the stability. Nodes with a lower degree have a significantly higher relative distance. There is not a lot of data for nodes with higher degrees but all of these edges have a very low relative distance. This means that GNNs also are more stable to perturbations around nodes with a high degree.

5 Responsible Research

All of the code used while researching this project can be found in the project repository[1]. The code contains sufficient documentation for others to reproduce and expand

[1]The project repository can be found at https://github.com/arg3t/TUDelft_CSE3000.

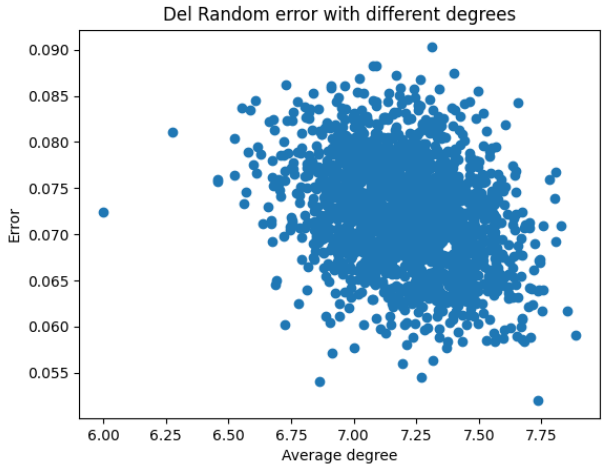

Figure 9: Relative distance of Delete random with 100 edge deletions

upon this research. The dataset is well-known in the graph machine-learning community and has been used in several papers. The existence of hidden bias in the dataset is not expected.

All the experiments are repeated multiple times to gain more confidence in the results. In the visualisations, the outliers are not depicted for clarity but are incorporated in the mean and median.

We investigated the behaviour of GNNs when perturbations are introduced with the goal of improving the performance of GNNs ultimately. This research would also aid malicious actors trying to attack the model. By intentionally supplying fabricated data the model is negatively influenced. This is done to destabilise the model or make the model not detect malicious inputs.

Understanding how different types of topological perturbations affect stability will inform the design of adversarial attacks and more importantly the design of adversarial defence. Adversarial defence is the field concerned with researching ways to build models resistant to adversarial attacks. According to the CUDOS principle, all research results are public property and should be published as soon as possible. We believe that GNNs can be made safer and more trustworthy by making these results public since researchers can create more robust models.

6 Conclusion

In this paper, we have investigated the impact of different topological perturbations on the impact of Graph Neural Networks. To do this a model was trained on the Cora dataset and then the data was perturbed with various strategies. The model's performance on the perturbed data was measured using the accuracy, the relative distance of the perturbed and unperturbed output, and the relative distance normalised

by the norm of the error. Additionally, the influence of the perturbation intensity and the degree of affected nodes was investigated.

Our main results are as follows: The stability is very dependent on the degree of adjacent nodes. Deleting edges is often less impactful than adding new edges since nodes adjacent to existing edges, on average, have a higher degree. The relation between the intensity of the perturbations is linear; GNNs are very resistant to large amount of perturbations.

The results in this paper lead to a better understanding of the behaviour of GNNs and their weaknesses. In situations where the behaviour of perturbations is known beforehand the performance of GNNs can be more predictable. Especially in the field of adversarial defence knowledge of the stability is essential.

For further research into this topic we recommend considering more strategies and graphs. Specifically, situations that are more likely to occur in real world applications. In the case of adversarial attacks the perturbations would try to maximise the error and in some context the graph might only grow and add new edges.

References

[1] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks," *CoRR*, vol. abs/2003.03777, 2020.

[2] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Transactions on Signal Processing*, vol. 68, p. 5680–5695, 2020.

[3] A. Daigavane, B. Ravindran, and G. Aggarwal, "Understanding convolutions on graphs," *Distill*, vol. 6, Aug 2021.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.

[5] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018.

[6] H. Kenlay, D. Thanou, and X. Dong, "Interpretable stability bounds for spectral graph filters," *CoRR*, vol. abs/2102.09587, 2021.

[7] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," 2019.

[8] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *CoRR*, vol. abs/1603.08861, 2016.

[9] X. Liu, Y. Zhang, M. Wu, M. Yan, K. He, W. Yan, S. Pan, X. Ye, and D. Fan, "Revisiting edge perturbation for graph neural network in graph data augmentation and attack," 2024.

7 Appendix: Additional Plots

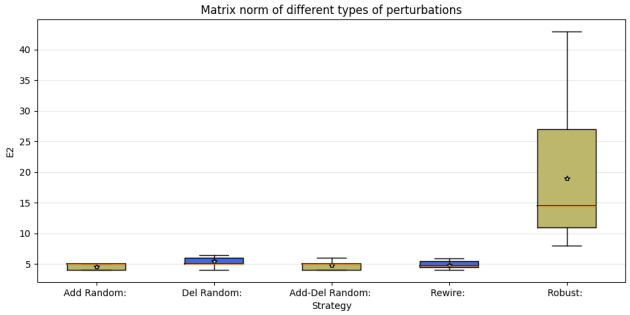

Figure 10: Norm of the error for different strategies

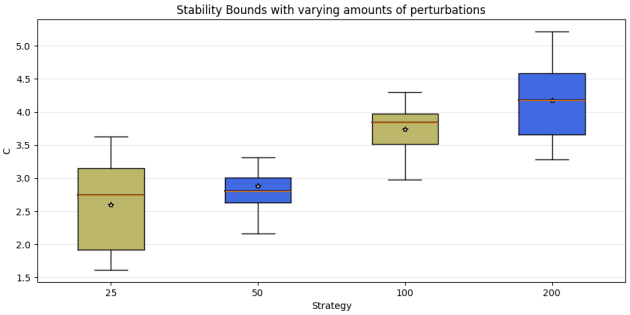

Figure 11: Stability bound with varying amount of perturbations

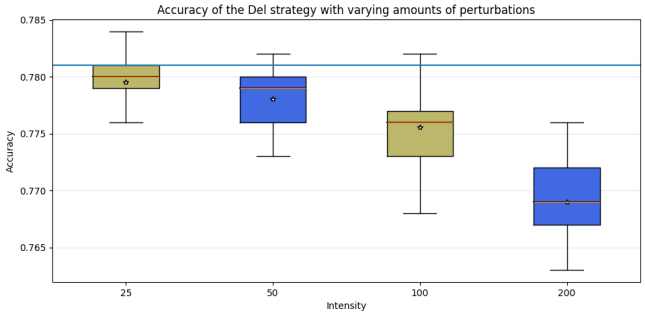

Figure 12: Accuracy of Delete Random with varying amount of perturbations

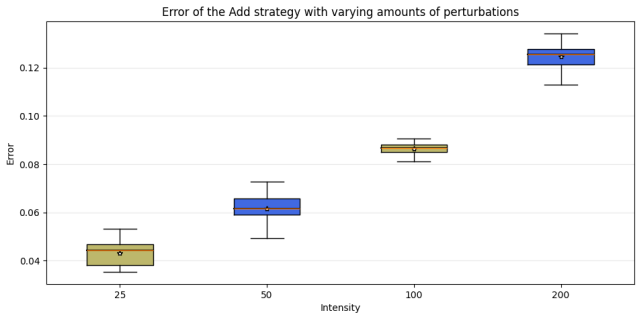

Figure 13: Relative distance of Delete Random with varying amount of perturbations

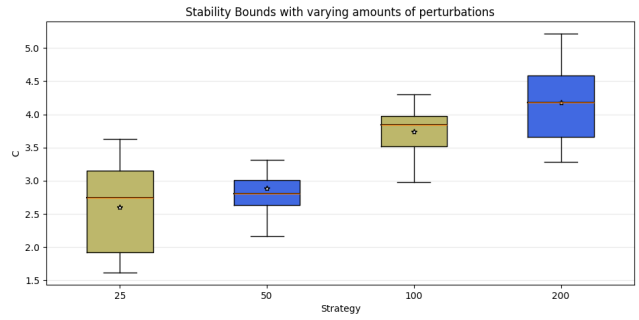

Figure 14: Stability bound of Delete Random with varying amount of perturbations