Perturbation Sensitivity of GNNs

Eric Winsor, winsor@stanford.edu

***Abstract*— This paper investigates the sensitivity of GNNs to various perturbations (edge additions and deletions) of the graph on which they were trained in the context of node classification. A theoretical framework is developed for discussing sensitivity of a GNN to certain kinds of perturbations. A number of factors that determine sensitivity are proposed. Experimental results that demonstrate the effects of these factors are given. These results showcase trade-offs between robustness and expressivity of GNNs, and show that robustness of node classification depends on factors such as node degree, topology and locality of node classes, type and placement of edge perturbation, and GNN depth. The results open up many possible avenues of inquiry into how different properties of graphs and GNNs affect sensitivity and how sensitivity relates to the expressive power of GNNs.**

I. INTRODUCTION

In the past few years, Graph Neural Networks (GNNs) have developed at an astonishing rate and achieved state of the art performance in many tasks [5]. Graph Convolutional Networks (GCNs) were a notable early architecture for GNNs [2]. GCNs were shown to have a significant advantage over previous state of the art node classification techniques. Since the debut of GCNs, a vast array of new GNN architectures have been proposed. One such architecture, GraphSage, achieved state of the art results on node classification and introduced the ability to learn representations of nodes across multiple graphs [1].

As they have been developed, GNN architectures have become ever more complicated. Notably, the Graph Isomorphism Network (GIN) is one of the most powerful GNNs to date, demonstrating state of the art performance on graph classification tasks [6]. GINs are notable for their foundation in a theoretical study of injective multiset functions and the Weisfeiler-Lehman (WL) graph isomorphism test. GINs have been proven to be as powerful as the WL isomorphism test, and empirical study has demonstrated that they are able to tap into a significant amount of this distinguishing power.

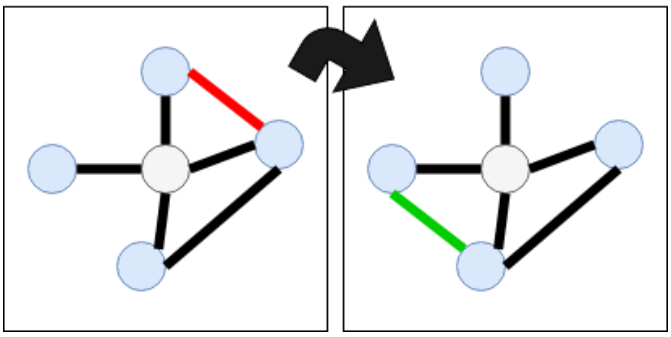

Fig. 1. Example of a (1,2)-perturbation about a node. The red edge (left) is deleted and the green edge (right) is added.

Despite these promising results, the inner workings of GNNs are not well understood. In fact, a recent study has shown that, in some cases, highly simplified GNN architectures can achieve state of the art performance on node classification tasks [4]. Simplified Graph Convolution (SGC) linearizes the aggregation step of GCNs, leading to huge speedups without significant performance degradation. Such a result suggests that modern GNN architectures may be more complicated than necessary; or at the very least, more research is needed to understand how to best utilize the characteristics of various GNN architectures.

Furthermore, the additional complexity may hurt the robustness of GNNs, since the resulting optimization landscape is not well understood. In fact, it has already been demonstrated that GNNs are vulnerable to adversarial attacks [7]. One such attack is node misclassification. To attack a GNN, a linearized version of it (similar to SGC) is constructed. Then, a set of edges is selected for removal in order to maximize the chance of misclassifying the node under this linearized model. The attack is then tested on the original GNN. With fairly high probability, the attack would work.

This result suggests that the seemingly diverse collection of GNN architectures may share many core characteristics.

Another challenge in using and understanding GNNs is handling the trade-off between capturing local and global structural features. As the depth (number of hops) of a GNN increases, more global information is captured. However, local information gets washed out by this global information, leading to a reduction in node classification performance. This is known as oversmoothing. Oversmoothing has been recently studied in detail, both theoretically and experimentally [3] [10]. GCNs lose expressive power (the ability to distinguish different neighborhoods) exponentially with depth. In order to improve node classification with more layers, it is necessary to utilize some form of residual connection that preserves information from early layers.

In light of these prior results, it is natural to wonder how sensitive GNNs are to local structure. Specifically, if edges in the neighborhood of a node are randomly added and removed, is the classification of that node significantly affected? If the classification is not affected, this suggests that GNNs are not really capturing fine structure, but instead memorizing the nodes that are in the neighborhood of a node. If classification is affected, then GNNs are learning the fine structure of a node's neighborhood.

This paper explores the sensitivity of GNNs to perturbations (edge additions and deletions) of the graph they are run on. We lay out a theoretical framework for analyzing a large class of perturbations that are relevant to node classification. We discuss factors that affect the sensitivity of GNNs to perturbations. Lastly, we give experimental results that demonstrate how sensitivity is affected by the factors we laid out. These results give insight into how much GNNs rely on fine local structure for node classification, as opposed to broad statistical features. Included in this is an observation that increasing the depth (hop count) causes the sensitivity of a GNN to decrease. This, together with prior results [3] [**?**], demonstrates that expressivity and sensitivity are closely linked and must be traded-off against one another when choosing GNN depth (for GNNs without residuals). Our results point the way to further investigations into how noise affects GNNs under various conditions.

II. RELATED WORK

There is no work on the effects of nonadversarial perturbations on GNNs. However, there has been some work on designing adversarial attacks on GNNs. [7] developed the first ever adversarial attacks on GNNs. These attacks altered node classifications by changing a small number of edges close to a target node and proved to be quite effective, demonstrating that GNNs are quite vulnerable to malicious adversaries. These attacks could be designed to only use edges outside the 1-hop neighborhood of a node, or even work when only a fraction of the overall graph structure is known (under imperfect information). Furthermore, attacks were demonstrated for both before and after training.

[8] used meta gradients to design nontargeted attacks. That is, these attacks could be used to poison the training process and produce a model whose overall classification accuracy was severely reduced. Importantly, this demonstrated that the overall classification accuracy of GNNs is not very robust. There exist relatively small sets of edges which, when added or removed, completely change the classification results for the whole graph. This motivates the question of how common such sets are, and whether classification is robust to nonadversarial perturbations.

[9] develops robustness certificates and a robust training procedure. The authors propose a way of verifying that GNNs have a certain level of worst case robustness to error, and propose a way of training GNNs to achieve this robustness. While this robustness notion is useful for protection against adversarial attacks, it does not necessarily give a way of optimizing robustness in a nonadversarial context. Also, this study does not investigate the general architectural features of GNNs that may make them more or less robust to perturbations, adversarial or otherwise.

[3] theoretically proves that, under certain assumptions, GNNs exponentially lose expressive power as the number of aggregation layers goes to infinity. This is interesting to the question of

robustness because the result suggests that GNNs with a large number of aggregation layers lose fine details of nodes, so they may be more robust. This creates evidence for an expressivity/robustness tradeoff to manifest in choosing an appropriate number of aggregation layers.

Lastly, [10] investigates oversmoothing. The authors introduce a metric known as MAD (mean average distance), which measures how smooth a representation (mapping from nodes to vectors) of a graph is. This metric is defined to be the average of the mean cosine difference from a node to other nodes (taken over some set of target node pairs). They also define an information-to-noise ratio to measure how much useful information is captured by a node's vector during classification. The authors show that beyond a certain optimum depth (dependent on the GNN architecture and input data) performance decreases rapidly. They explain this by demonstrating that greater depth increases the smoothness of a representation and sharply decreases the information-to-noise ratio. This provides strong intuition for the idea that depth has a large effect on how sensitive GNNs are to local structure.

III. DEFINITIONS AND MATHEMATICAL BACKGROUND

For our definitions of GNNs, we follow the notation of [6]. Define a graph G to be a pair (V, E) of a set, V, of vertices and a set, E, of edges. Given a graph G and a node feature vector, X_v, the kth layer of a GNN, which outputs a feature vector $h_v^{(k)}$, can be defined as

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) \tag{1}$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right). \tag{2}$$

In general, AGGREGATE and COMBINE are parameterized, and the GNN is trained by fitting those parameters. Note that these functions depend both on X_v and G, and we will be interested in the effect of changes in G on these functions.

We are interested in the application of GNNs to node classification. In node classification, a graph G is given and some subset of its nodes are labelled with y_v where $v \in V$. A GNN is then used to learn a corresponding mapping of nodes to vectors, h_v, such that some classifier, f, yields $f(h_v) = y_v$ for labelled nodes. Note that the flexibility of how GNNs are defined allows for a GNN trained on a particular graph to be applied (with varying success) to any other graph. Importantly, a GNN trained on a graph G can still be applied to that graph G after adding or removing edges. This will allow us to measure the sensitivity of a given GNN to changed in the structure of G.

We will be specifically interested in GCNs, which were introduced by Kipf and Welling [2]. GCNs are commonly defined by defining the composition of the COMBINE and AGGREGATE functions as a matrix multiplication followed by an activation function. Specifically, let A be the adjacency matrix of an undirected graph G, and let I be the identity matrix. We define $\tilde{A} = A + I$ and $\tilde{D} = \sum_j \tilde{A}_{ij}$. Let $H^{(k)}$ be the matrix of feature vectors, $h_v^{(k)}$. Then a GCN is defined by

$$H^{(k+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(k)}W^{(k)}\right), \tag{3}$$

where $W^{(k)}$ is a learned weight matrix and σ is an activation function (e.g. ReLU).

A. Graph Perturbations

We now introduce a new notion of graph perturbations, which is central to our investigations. Let $G = (V, E)$ be a graph and let v be a fixed vertex of G. Define a probability distribution $\rho_{G,v}$ on the set of graphs on the vertex set V. We will say that a graph H on V is a (k, ℓ) perturbation of G if the edge set of H differs (by addition or deletion) from that of G by at most ℓ edges, and all of these differing edges are between vertices that are within k hops of v. We will say that $\rho_{G,v}$ is a (k, ℓ)-perturbation distribution (or just a (k, ℓ)-distribution) if all graphs with nonzero probability under $\rho_{G,v}$ are (k, ℓ)-perturbations of G. An example of a $(1, 2)$-perturbation is shown in Figure 1.

In this framework, one could say that [9] studies worst-case performance of node classification under (k, ℓ)-perturbations, where k is the depth of the neural net (in their case, this depth is 3), and ℓ is defined by some perturbation budget. While this prior study was focused on worst-case,

adversarial perturbations, our study will be focused on randomized perturbations.

Let a GNN with a classifier be a function $f(G, v, h_v; \theta)$ into node class labels, where G is a graph, v is a vertex of G, h_v is the feature vector of v, and θ is the parameters of f. Now consider a (k, ℓ) distribution $\rho_{G,v}$. The perturbation sensitivity at v of our GNN with respect to $\rho_{G,v}$ can then be defined as the probability that $f(G, v, h_v; \theta) \neq f(G', v, h_v)$ where G' is drawn from $\rho_{G,v}$. In other words, the perturbation sensitivity of our GNN at v is the probability that the classification of v changes. We may then average the perturbation sensitivity of a GNN over a set of vertices to get its perturbation sensitivity with respect to that set. In our experiments, our perturbation distribution will typically be defined by randomly sampling ℓ-many valid edges and toggling them (adding or removing them).

IV. THEORETICAL CONSIDERATIONS

Rigorously and formulaically analyzing all factors that might play into perturbation sensitivity is beyond the scope of existing theory. However, we can describe a number of factors that could conceivably affect it and offer a heuristic explanation.

A. Number of Edge Modifications

The most obvious possible factor is the number of edge modifications applied to a graph. Adding or deleting many edges at random in the neighborhood of a node v will significantly alter the neighborhood structure. Thus, if the GNN being used is sensitive to local structure, and the local structure is changed significantly, there should be a high probability of changes in node classification.

B. Node Degree and Size of Neighborhood

Nodes of higher degree should be less sensitive to perturbations than nodes of low degree. Since each neighbor will factor into aggregate step (which is usually mean, max, or min), changes in the edges of a single neighbor will be less noticeable when there are many other neighbors. If there is a particular neighbor that plays a crucial role in classification, that neighbor is less likely to be affected by a perturbation with a fixed number of edge modifications if the degree of the vertex of interest is high (it is simply less likely to have one of its edges chosen). Similar logic applies to the k-hop neighborhood when considering (k, ℓ)-perturbations. The larger the neighborhood is, the less of a role that any one edge should, on average, play in classifying a vertex.

C. Locality and Topology of Node Classes

If node classes are well clustered in a graph (i.e. a vertex is much more likely to share an edge with a vertex of a similar class than with one of a different class), then perturbation sensitivity should be reduced. In the extreme case where node classes form disjoint connected components, a GNN can simply propagate node classifications from classified nodes to unclassified nodes. Thus, the GNN would not need to be sensitive to local structure whatsoever, so long as changes to the structure preserve connected components. Sensitivity to local structure in such a case might actually suggest that the GNN has been overfitted. Most real world data should have a large amount of locality. On the other hand, if one were to generate a random graph with random class labels, it might also be expected that the GNN would have low sensitivity to local structure (since the local structure is completely random). Thus, a GNN is going to have the most perturbation sensitivity when the class labels depend heavily on fine structure and patterns in the graph.

D. Location of Edge Modifications

Edge modifications that occur multiple hops away from a node of interest are less likely to effect classification than edge modifications that occur close to the node. If a GNN uses k-many hops, then every k-hop path ending at the vertex of interest will have an effect on its classification. An edge that is closer to the vertex of interest will have many more valid k-hop paths passing through it, so it will have more weight in the final classification. This is noted in the adversarial case in [7], where the authors focus on making edge changes close to a node, but in one case restrict themselves from changing edges directly connected to a target node. It is noted that not allowing modification to direct connections makes it much harder to find a good adversarial perturbation. This is additional evidence for the hypothesis that edges that a far

from a vertex have less influence on its classification.

E. Edge Additions Or Deletions

Edges being added may have a different effect than edges being deleted. If class labels have locality, then edge additions are more likely to strengthen that locality, whereas edge deletions are more likely to weaken it. Thus, one might hypothesize that restricting to edge deletions would increase perturbation sensitivity, whereas restricting to edge additions would decrease it.

F. GNN Depth

A GNN with greater depth should generally be less perturbation sensitive than one with lesser depth. Such a result can roughly be inferred from [3] and [10]. In the limit of infinite depth, node embeddings become very similar to each other and become primarily dependent on the spectrum of the graph. A small number of random edge modifications will have very little effect on this, so GNNs with greater depth should be less perturbation sensitive.

V. EXPERIMENTAL RESULTS

For experiments, we used a GCN and the Cora dataset. We start with the basic setup of [2], using a 0.5 dropout rate, $5 \cdot 10^{-4}$ L2 regularization, and 16 hidden units. Since our goal is not to measure accuracy, but perturbation sensitivity, we fix the number of training epochs to 200 (rather than choosing an optimally accurate epoch).

Cora is a citation network, where each node represents a document and each directed edge represents a citation. Each document is given a sparse bag-of-words feature vector. We treat the graph as undirected. There are 7 document classes and 1,433 features. There is a total of 2,708 nodes and 5,429 edges. We use the same dataset splits as in [2]. Note that Cora has a power law degree distribution and nodes tend to be connected to nodes with similar classifications.

For each experimental run, we measure perturbation sensitivity with respect to some perturbation scheme. For each measurement, we train the GCN from scratch and then select a random node from what would normally be the test set of the GCN. We then sample the perturbation distribution about that node 1,000 times and count the number of times the node classification changes as a result of changing edges in the graph. We repeat this procedure with 1,000 different nodes, retraining the GCN each time to ensure that the results are independent of any pathologies of an individual training result.

A. Baseline Experiment and Bimodality

We begin by using the setup described above with a $(1, 1)$-perturbation scheme. Specifically, we choose a random potential edge in the egonet of the chosen node and toggle it (adding or removing it). We then count up the number of times the node classification changed as a result. The results are shown in Figure 2.

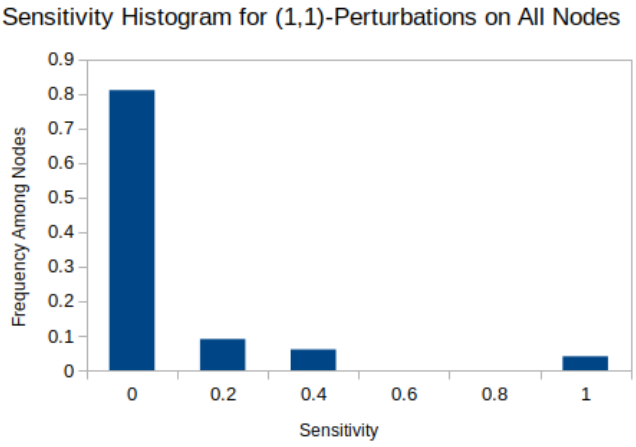

Fig. 2. Histogram of sensitivities among all nodes using a $(1, 1)$-perturbation scheme and baseline setup. Note that the bar at e.g. 0.2 represents the range $0 < x <= 0.2$.

Over 80% of sampled nodes never changed classification, despite being perturbed 1,000 times. This can be explained by our previous observation about locality of node classes. Similar document classes are more likely to cite one another, so the node classes in this dataset have a significant degree of locality. This is solid evidence that a GCN does not (or at least does not need to) pay attention to very fine-grained local structure in the graph. Of course, this experiment alone is evidence only for a weaker hypothesis: the local structure captured by a GCN has some redundancy. That is, the vast majority of vertices are agnostic to single edge changes because the GCN is capturing

features that are encoded by multiple edges. On the other hand, we are now confident that the GCN in this experiment is definitely not encoding all of the local structure of the graph.

Another noteworthy result from this experiment is that about 5% of nodes had a 100% sensitivity. These are all nodes of degree 1. Their egonet is a single connected pair of nodes, so toggling an edge necessarily means cutting the only edge in the egonet. The GCN is then forced to classify the node based on only internal features. Based on the outcome of this classification, the sensitivity will be either 0 or 1 under this perturbation. Overall, about 70% of degree 1 nodes have sensitivity 0 and 30% have sensitivity 1. Thus, it seems that most degree 1 nodes have their classification reinforced by their neighbor, which is consistent with the locality of classes in this dataset.

The nodes sensitivity strictly between 0 and 1 mostly have degree less than 5. This is coincides with our hypothesis that higher degree nodes have lower sensitivity. Of course, it should be noted that the Cora dataset has a power law distribution, so high degree nodes have a low sample probability to begin with, and more experiments will be needed in order to determine the relationship between node degree and sensitivity.

As a final note, one may observe that the sensitivity histogram looks like a power law distribution (ignoring the bump at 1). The histograms resulting from most of our experiments look like this, so we will not include them. Instead, we will generally be analyzing average sensitivities and the fraction of sensitivities that are nonzero.

B. Node Degree

We now consider similar experiments to that of the baseline. However, we restrict ourselves to sampling nodes of degree at least 10, and again with only nodes of degree at least 20. We consider using 1 or 100 edge modifications (we use different numbers of modifications because of the vastly different scales of neighborhood sizes). We measured the fraction of nodes with zero sensitivity, and the average sensitivity of nodes with nonzero sensitivity. The results are shown in Figure 3.

As we expected, sensitivity decreases as degree increases. A larger neighborhood causes modified edges to be less important overall, since there are more edges, and the randomly placed edges are more diffuse (there cannot be significant sparsification or densification in any area). We can conclude that GCNs are less sensitive to the local structure of high degree nodes than that of low degree nodes.

Min. Deg.	# Mod.	Frac. Zeros	Avg. Sens.
1	1	0.81	0.39
10	1	0.92	0.03
10	100	0.78	0.28
20	100	0.95	0.06

Fig. 3. Results of $(1, \ell)$-perturbations with 1 or 100 edge modifications for nodes with degree at least 1, 10, or 20. Shown is the fraction of nodes with zero sensitivity and the average sensitivity not including zeros.

C. Number of Edge Modifications

We now restrict ourselves to sampling nodes of degree at least 10 and allow the number of modifications to vary. We measure the results of our experiment using 1, 10, and 100 modifications. We measured the fraction of nodes with zero sensitivity, and the average sensitivity of nodes with nonzero sensitivity. The results are shown in Figure 4.

Num. Mod.	Frac. Zeros	Avg. Sens.
1	0.92	0.03
10	0.83	0.15
100	0.78	0.28

Fig. 4. Results of $(1, \ell)$-perturbations with 1, 10, or 100 edge modifications for nodes with degree at least 10. Shown is the fraction of nodes with zero sensitivity and the average sensitivity not including zeros.

Unsurprisingly, increasing the number of edge modifications decreases the fraction of nodes with zero sensitivity and increases the average sensitivity of nodes with nonzero sensitivity. With more changes to the local structure, the classification becomes more random. This shows that a GCN is at least somewhat sensitive to changes in local structure. Of course, even with 100 modifications almost 80% of nodes had no change in classification, so this experiment also suggests that the GCN is not nearly as sensitive to local structure as an arbitrary node classifier theoretically could be.

D. Edge Additions Or Deletions

We now consider 3 experiments using $(1, 100)$-perturbations on nodes of degree at least 10 (sampling 100 random edges in the egonet to be changed). In the first, we allow for additions or deletions of edges. In the second, we allow for only deletions (we sample 100 modifications and only perform the deletions, meaning we have a smaller effective number of modifications). In the third, we allow for only additions (using the same scheme as with deletions). We measured the fraction of nodes with zero sensitivity, and the average sensitivity of nodes with nonzero sensitivity. The results are shown in Figure 5.

	Frac. Zeros	Avg. Sens.
Additions	0.94	0.71
Deletions	0.49	0.19
Both	0.78	0.28

Fig. 5. Results of $(1, 100)$-perturbations (additions, deletions, or both) for nodes with degree at least 10. Shown is the fraction of nodes with zero sensitivity and the average sensitivity not including zeros.

The results are consistent with our earlier theoretical discussion. Because of the locality of class labels, restricting to deletions actually increases the number of nodes with nonzero sensitivity, despite the decrease in the effective number of modifications due to our experimental setup. Because most nodes are connected to nodes with similar class labels, deleting a large number of edges makes it hard to "pick up" the featural similarities between a node and its neighbors. On the other hand, using only additions almost tripled the number of nodes with 0 sensitivity. Since the nearby nodes are likely to be similar, adding more connections simply strengthens this perceived similarity, and thus does not usually have much of an effect on classification.

The difference in averages is due to additions yielding a sensitivity distribution with a heavy tail near 1, whereas deletions yielded a thinner tail. Note that deletions reduce the number of edges in the neighborhood of a node, which will generally already be sparse, so after deleting some edges, the strength of dependency on the neighboring nodes is likely to be fairly similar to what it was before. However, additions significantly densify the 1-hop neighborhood of a node, so they can cause a thresholding phenomenon. The strength of dependence on its neighbors will increase dramatically, so if its class label originally differed from that of the majority of its neighbors, it may suddenly switch to match its neighbors due to the strong new influence. This explains the trend in the average sensitivities.

In summary, large numbers of deletions tend to cause a weaker, more random error in classification, whereas large numbers of additions tend to cause strong, fairly deterministic errors in classification. These behaviors suggest that the GCN is fairly sensitive to "who" is in the neighborhood of a node and how numerous the connections are, but not as sensitive to the exact structure of these connections.

E. Location of Edge Modifications

We now consider 3 experiments using a GCN of depth 4 (with 16 hidden units at each hidden level). We consider random $(k, 100)$ perturbations for $k = 2, 3, 4$ (choosing 100 random edges in the k-hop neighborhood of a node) with no restrictions on node degree. We measured the fraction of nodes with zero sensitivity, and the average sensitivity of nodes with nonzero sensitivity. The results are shown in Figure 6.

Hops	Frac. Zeros	Avg. Sens.
2	0.72	0.28
3	0.59	0.21
4	0.46	0.11

Fig. 6. Results of $(k, 100)$-perturbations for nodes with any degree using a GCN depth of 4 and modifications within 2, 3, or 4 hops of the chosen node. Shown is the fraction of nodes with zero sensitivity and the average sensitivity not including zeros.

We can see that as we allow for edges to get farther away, the fraction of nodes that have zero sensitivity decreases, but the average sensitivity decreases by a march larger proportion. Essentially, the sensitivity distribution is spreading out and thinning at the same time. This can be explained as follows.

Firstly, allowing for more hops increases the set of nodes under consideration. Importantly, nodes that are 3 or 4 hops away may be very different from the chosen node. With 100 available edge

modification, there is ample opportunity for connecting far away nodes so that they are only 1 or 2 hops away from the chosen node. This means that there is a nontrivial chance of changing the broad structure of the egonet or 2-hop neighborhood of the chosen node (e.g. significantly changing the relative proportions of node classes). Thus, with more hops, the proportion of nodes with 0 sensitivity naturally drops a bit because nodes surrounded by similar nodes (at the 1 or 2-hop scale) could now connect to different nodes.

Secondly, allowing for more hops dilutes the average effectivess of edges. As discussed in our theoretical section, far away edges will generally have less influence on the classificaiton of a chosen node. By increasing the allowable distance of edges from the chosen node, the probability of changing a node's classification decreases because the chosen edges tend to be less influential. This causes the average sensitivity to decrease significantly, even as more nonzero sensitivities appear as a result of the previously discussed effect.

In total, nodes, which at the 1 or 2-hop level had their sensitivity "thresholded out" to 0 because they were surrounded by similar nodes, are able to move their sensitivity a little bit away from 0. At the same time, nodes with nonzero sensitivity have their sensitivities decrease due to the decrease in average edge influence. Hence, the spike at 0 sensitivity expands out while the overall distribution contracts towards 0, yielding the observed effect. In the limit of a very large graph and large number of hops, we would expect this sensitivity to collapse towards 0.

F. GCN Depth

We now consider 4 experiments using GCN depths of 2, 3, 4, and 10 (with 16 hidden units at each hidden level). Note that we have fixed the number of epochs at 200, so the deeper GCNs may be suboptimally trained. We leave the investigation of how training optimality affects sensitivity to future work. We run our experiments considering nodes of degree at least 10 using (1,100)-perturbations. We measured the fraction of nodes with zero sensitivity, and the average sensitivity of nodes with nonzero sensitivity. The results are shown in Figure 7.

Depth	Frac. Zeros	Avg. Sens.
2	0.78	0.28
3	0.73	0.15
4	0.73	0.22
10	1	N/A

Fig. 7. Results of $(1, 100)$-perturbations for nodes with degree at least 10 using GCN depths of 2, 3, 4, or 10. Shown is the fraction of nodes with zero sensitivity and the average sensitivity not including zeros.

At depths of 2, 3, and 4, the distributions seem to spread out (leading to a smaller fraction of zeros for 3, and then a larger average sensitivity for 4). This is similar to the trend from the previous experiment on the effect of edge modification location, although here the effect is much smaller and less clear cut. Thus, this trend is too weak to make any claims about. On the other hand, at depth 10, all sensitivities are 0. Thus, beyond low depths, there is a steep drop off in sensitivity. This harmonizes with the results of [3] and [10], which showed that expressivity decreases rapidly with depth. In short, increasing depth causes a decrease in expressivity but an increase in robustness (decrease in sensitivity). Thus, we have an interesting demonstration of a trade-off between the distinguishing power of GNNs and their robustness to noise.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

We have developed a theoretical framework for analyzing the sensitivity of GNNs to various types of noise in the setting of node classification. We have given experimental results that show how sensitivity is dependent on factors such as node degree, size, location, and type of perturbation, topology and locality of node classes, and depth of GNN architecture. Additionally, we have observed how our notion of sensitivity relates to the expressivity of a GNN, its ability to distinguish different structures. This connection was revealed in the varying effects of edge additions and deletions, node degree, and GNN depth during our experiments. In general, it seems that sensitivity and expressivity should trade off against one another. Further work could be done to better understand this trade-off.

In addition to connecting expressivity and sensitivity, future experiments could be performed to

understand how sensitivity works on graphs that do not have a power law degree distribution or on graphs that have less class locality. Additionally, GNNs other than the GCN could be analyzed, and sensitivity could be used as a point of comparison for different GNN architectures. Indeed, many more experiments could be performed analyzing the interactions of different graphs, both real-world and synthetic, and different GNN architectures, to further explicate the factors that affect sensitivity.

Another future route could be expanding this work to graph classification. A challenge in this would be determining what constitutes a "small" perturbation because graph classification considers a whole graph, instead of the neighborhood of a single node. One might expect that graph classification is less sensitive to small localized perturbations, but this would be very interesting to test.

Lastly, it would be interesting to mathematically investigate sensitivity. Specifically, finding and proving formulaic relations between sensitivity, node degree, types of perturbations, and GNN depth, even using simple models, could be very fruitful for understanding where sensitivity "comes from". That is, how much, and in what ways, does sensitivity depend on all of these different factors? A mathematical result might even shed light on the inner workings of GNNs and lead to a better overall understanding of how they perform classification tasks, and where they might fail.

This paper has only scratched the surface of possible investigations into perturbation sensitivity and its underlying factors. The results here provide a strong foundation on which to build further research. Hopefully, future insights into perturbation sensitivity will yield a deeper understanding of the complicated inner workings of GNNs.

REFERENCES

[1] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems* (pp. 1024-1034).

[2] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907.*

[3] Oono, K., & Suzuki, T. (2019). Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. *arXiv preprint arXiv:1905.10947.*

[4] Wu, F., Zhang, T., Souza Jr, A. H. D., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153.*

[5] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2019). A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596.*

[6] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks?. *arXiv preprint arXiv:1810.00826.*

[7] Zügner, D., Akbarnejad, A., & Günnemann, S. (2018, July). Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2847-2856). ACM.

[8] Zügner, D., & Günnemann, S. (2019). Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412.*

[9] Zügner, D., & Günnemann, S. (2019, July). Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 246-256). ACM.

[10] Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2019). Measuring and Relieving the Over-smoothing Problem for Graph Neural Networks from the Topological View. *arXiv preprint arXiv:1909.03211.*