What Do GNNs Actually Learn? Towards Understanding their Representations

Giannis Nikolentzos
University of Peloponnese, Greece
nikolentzos@uop.gr

Michail Chatzianastasis
LIX, École Polytecnique, IP Paris, France
mixalisx97@gmail.com

Michalis Vazirgiannis
LIX, École Polytecnique, IP Paris, France
mvazirg@lix.polytechnique.fr

Abstract

In recent years, graph neural networks (GNNs) have achieved great success in the field of graph representation learning. Although prior work has shed light on the expressiveness of those models (i. e., whether they can distinguish pairs of non-isomorphic graphs), it is still not clear what structural information is encoded into the node representations that are learned by those models. In this paper, we address this gap by studying the node representations learned by four standard GNN models. We find that some models produce identical representations for all nodes, while the representations learned by other models are linked to some notion of walks of specific length that start from the nodes. We establish Lipschitz bounds for these models with respect to the number of (normalized) walks. Additionally, we investigate the influence of node features on the learned representations. We find that if the initial representations of all nodes point in the same direction, the representations learned at the k-th layer of the models are also related to the initial features of nodes that can be reached in exactly k steps. We also apply our findings to understand the phenomenon of oversquashing that occurs in GNNs. Our theoretical analysis is validated through experiments on synthetic and real-world datasets.

1 Introduction

Graphs arise naturally in a wide variety of domains such as in bio- and chemo-informatics [1], in social network analysis [2] and in information sciences [3]. There is thus a need for machine learning algorithms that can operate on graph-structured data, i. e., algorithms that can exploit both the information encoded in the graph structure but also the information contained in the node and edge features. Recently, graph neural networks (GNNs) emerged as a very promising method for learning on graphs, and have driven the rapid progress in the field of graph representation learning [4].

Even though different types of GNNs were proposed in the past years, message passing models undoubtedly seem like a natural approach to the problem. These models, known as message passing neural networks (MPNNs) [5], employ a message passing (or neighborhood aggregation) procedure where each node aggregates the representations of its neighbors along with its own representation to produce new updated representations. For graph-related tasks, MPNNs usually apply some permutation invariant readout function to the node representations to produce a representation for the entire graph. The family of MPNNs has been studied a lot in the past few years, and there are now available dozens of instances of this family of models. A lot of work has focused on investigating the expressive power of those models. It was recently shown that standard MPNNs are at most as powerful as the Weisfeiler-Leman algorithm in terms of distinguishing non-isomorphic graphs [6, 7].

G. Nikolentzos et al., What Do GNNs Actually Learn? Towards Understanding their Representations. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

The recent success of GNNs put graph kernels, another approach for graph–based machine learning, into the shade. Unlike GNNs, graph kernels generate representations (implicit or explicit) that typically capture some substructure of graphs [8]. Such substructures include random walks [9, 10], shortest paths [11] and subgraphs [12, 13]. Therefore, the properties and the graph representations produced by most graph kernels are fully-understood. This is not however the case for MPNNs since, despite the great activity in the field, still little is known about the properties of graphs that are captured in the representations learned by those models.

In this paper, we fill this gap by studying the node representations learned by MPNNs. We first investigate what structural properties of graphs are captured in the learned representations of standard models. To study those representations, we capitalize on Lipschitz continuity, a standard tool for analyzing representations of neural network models and for assessing their robustness to perturbations [14, 15]. We show that when all nodes are annotated with the same features, both GAT [16] and DGCNN [17] embed all nodes into the same vector. Furthermore, we show that the representations that emerge at the k-th layer of GCN [18] and GIN [6] are related to some notion of walks of length k over the input graph. This suggests that MPNNs suffer from the following limitation: structurally dissimilar nodes can have similar (or even identical) representations at some layer k where $k > 1$. We also study the impact of node features on the learned representations. We show that if the initial features of all nodes point in the same direction, the node representations at the k-th layer of GCN and GIN are all related to the initial features of the nodes that can be reached in exactly k steps from the node. We finally study the problem of oversquashing [19] from the lens of our theoretical findings. We verify our theoretical analysis in experiments conducted on synthetic and real-world datasets. Our code is available at `https://github.com/giannisnik/gnn-representations`.

2 Related Work

While GNNs have been around for decades [20–22], it is only in recent years that the scientific community became aware of their power and potential. The increased scientific activity in the field led to the development of a large number of models [23–27]. Those models were categorized into spectral and spatial approaches depending on which domain the convolutions (neighborhood aggregations) were performed. Later, it was shown that all these models can be seen as instances of a single common framework [5]. These models, known as message passing neural networks (MPNNs), use a message passing scheme where nodes iteratively aggregate feature information from their neighbors. Then, to compute a representation for the entire graph, MPNNs typically employ some permutation invariant readout function which aggregates the representations of all the nodes of the graph. In the past few years, there have been proposed several extensions and improvements to the MPNN framework. Most studies have focused on the message passing procedure and have proposed more expressive or permutation sensitive aggregation functions [28–31], schemes that incorporate different local structures or high-order neighborhoods [32, 33], non-Euclidean geometry approaches [34], while others have focused on efficiency [35]. Fewer works have focused on the pooling phase and have proposed more advanced strategies for learning hierarchical graph representations [36, 37]. Note also that not all GNNs belong to the family of MPNNs [38–40], and that there exist models which process random walks sampled from the graph [41, 42] and which can mitigate MPNNs' common symptoms such as oversmoothing and oversquashing.

A considerable amount of recent work has focused on characterizing the expressive power of GNNs. Most of these studies compare GNNs against the WL algorithm and its variants [43] to investigate what classes of non-isomorphic graphs they can distinguish. For instance, it has been shown that standard GNNs are not more powerful than the 1-WL algorithm [6, 7]. Other studies capitalized on high-order variants of the WL algorithm to derive new models that are more powerful than standard MPNNs [7, 44]. Recent research has investigated the expressive power of k-order GNNs in terms of their ability to distinguish non-isomorphic graphs. In particular, it has been shown that k-order GNNs are at least as powerful as the k-WL test in this regard [45]. Various approaches have also been proposed to enhance the expressive power of GNNs beyond that of the WL test. These include encoding vertex identifiers [46], incorporating all possible node permutations [28, 47], using random features [48, 49], utilizing node features [50], incorporating spectral information [51], utilizing simplicial and cellular complexes [52, 53] and directional information [54]. It has also been shown that extracting and processing subgraphs can further enhance the expressive power of GNNs [55–57]. For instance, it has been suggested that expressive power of GNNs can be increased by aggregating the representations of subgraphs produced by standard GNNs, which arise from removing one or

Table 1: Neighborhood aggregation schemes of the four considered models.

Model	Update Equation
GCN	$\mathbf{h}_v^{(k)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{\mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}}{\sqrt{(1+d(v))(1+d(u))}} \right)$
DGCNN	$\mathbf{h}_v^{(k)} = f\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{d(v)+1} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)} \right)$
GAT	$\mathbf{h}_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)} \right)$
GIN-ϵ	$\mathbf{h}_v^{(k)} = \text{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k-1)} \right)$

more vertices from a given graph [58, 59]. The above studies mainly focus on whether GNNs can distinguish pairs of non-isomorphic graph. However, it still remains unclear what kind of structural information is encoded into the node representations learned by GNNs. Some recent works have proposed models that aim to learn representations that preserve some notion of distance of nodes [60], however, they do not shed light into the representations generated by standard models. The work closest to ours is the one proposed by Chuang and Jegelka [61], where the authors propose the Tree Mover's Distance, a pseudometric for node-attributed graphs, and study its relation to the generalization of GNNs. Our work is also related to the work of Xu et al. [62] where the authors use the concept of walks to define the effective range of nodes that any given node's representation draws from. However, while this work studies the range of nodes whose features affect a given node's representation, we focus on the exact node representations that are learned by the model. Finally, Yehudai et al. [63] capitalize on local computation trees and graph patterns similar to the ones studied in this paper to investigate the GNNs' ability to generalize to larger graphs.

3 Preliminaries

3.1 Notation

Let $\mathbb{N}$ denote the set of natural numbers, i. e., $\{1, 2, \ldots\}$. Then, $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ for $n \geq 1$. Let also $\{\!\{\}\!\}$ denote a multiset, i. e., a generalized concept of a set that allows multiple instances for its elements. Let $G = (V, E)$ be a (directed) graph, where V is the vertex set and E is the edge set. We will denote by n the number of vertices and by m the number of edges, i. e., $n = |V|$ and $m = |E|$. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ encodes the edge information in a graph. The element of the i^{th} row and j^{th} column is equal to 1 if there is an edge between v_i and v_j, and 0 otherwise. Let $\mathcal{N}(v)$ denote the the neighbourhood of vertex v, i. e., the set $\{u \mid (u, v) \in E\}$. The degree of a vertex v is $d(v) = |\mathcal{N}(v)|$. A walk in graph G is a sequence of vertices $v_1, v_2, \ldots, v_{k+1}$ where $v_i \in V$ and $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k$. We denote by $w_v^{(k)}$ the number of walks of length k starting from node v. Finally, let $\tilde{w}_v^{(k)}$ denote the sum of normalized walks of length k where each walk $(v_1, v_2, \ldots, v_k)$ is normalized as follows $1/\left((1+d(v_2))\ldots(1+d(v_{k-1}))\sqrt{(1+d(v_1))(1+d(v_k))}\right)$.

3.2 Message Passing Neural Networks

As already discussed, most GNNs can be unified under the MPNN framework [5]. These models follow a neighborhood aggregation scheme, where each node representation is updated based on the aggregation of its neighbors' representations. Let $\mathbf{h}_v^{(0)}$ denote node v's initial feature vector. Then, for a number K of iterations, MPNNs update node representations as follows:

$$\mathbf{m}_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\{\!\{ \mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}(v) \}\!\} \right)$$

$$\mathbf{h}_v^{(k)} = \text{COMBINE}^{(k)}\left(\mathbf{h}_v^{(k-1)}, \mathbf{m}_v^{(k)} \right)$$

where $\text{AGGREGATE}^{(k)}$ is a permutation invariant function. By defining different $\text{AGGREGATE}^{(k)}$ and $\text{COMBINE}^{(k)}$ functions, we obtain different MPNN instances. In this study, we consider the neighborhood aggregation schemes of four models, namely (1) Graph Convolution Network (GCN) [18]; (2) Deep Graph Convolutional Neural Network (DGCNN) [17]; (3) Graph Attention Network (GAT) [16]; and (4) Graph Isomorphism Network (GIN) [6]. The aggregation schemes of the four models are illustrated in Table 1. Note that for directed graphs, $\mathcal{N}(v)$ is a set that contains the in-neighbors of v.

For node-level tasks, the final node representations $\mathbf{h}_v^{(K)}$ can be directly passed to a fully-connected layer for prediction. For graph-level tasks, a graph representation is obtained by aggregating the final representations of its nodes: $\mathbf{h}_G = \text{READOUT}\Big(\{\!\{\mathbf{h}_v^{(K)} \mid v \in G\}\!\}\Big)$. The READOUT function is typically a differentiable permutation invariant function such as the sum or mean operator.

4 What Do MPNNs Actually Learn?

We next investigate what structural properties of nodes these four considered models can capture. Nodes are usually annotated with features that reveal information about their neighborhoods. Such features include their degree or even more sophisticated features such as counts of certain substructures [64] or those extracted from Laplacian eigenvectors [65]. We are interested in identifying properties that are captured purely by these models. Thus, we assume that no such features are available, and we annotate all nodes with the same feature vector or scalar.

Theorem 1. *Let $\mathcal{G} = \{G_1, \ldots, G_N\}$ be a collection of graphs. Let also $\mathcal{V} = V_1 \cup \ldots \cup V_N$ denote the set that contains the nodes of all graphs. All nodes are initially annotated with the same representation. Without loss of generality, we assume that they are annotated with a single feature equal to 1, i. e., $\mathbf{h}_v^{(0)} = 1 \; \forall v \in \mathcal{V}$. Then, after k neighborhood aggregation layers:*

1. *DGCNN and GAT both map all nodes to the same representation, i. e., $\mathbf{h}_v^{(k)} = \mathbf{h}_u^{(k)} \; \forall v, u \in \mathcal{V}$.*
2. *GCN maps nodes to representations related to the sum of normalized walks of length k starting from them:*

$$\left|\left|\mathbf{h}_v^{(k)} - \mathbf{h}_u^{(k)}\right|\right|_2 \leq \prod_{i=1}^{k} L_f^{(i)} \left|\left|\tilde{w}_v^{(k)} - \tilde{w}_u^{(k)}\right|\right|_2$$

   where $L_f^{(i)}$ denotes the Lipschitz constant of the fully-connected layer of the i-th neighborhood aggregation layer.

3. *Under mild assumptions (biases of MLPs are ignored), GIN-0 maps nodes to representations that capture the number of walks of length k starting from them:*

$$\left|\left|\mathbf{h}_v^{(k)} - \mathbf{h}_u^{(k)}\right|\right|_2 \leq \prod_{i=1}^{k} L_f^{(i)} \left|\left|w_v^{(k)} - w_u^{(k)}\right|\right|_2$$

   where $L_f^{(i)}$ denotes the Lipschitz constant of the MLP of the i-th neighborhood aggregation layer.

The above result highlights the limitations of the considered models. Specifically, our results imply that the DGCNN and GAT models encode no structural information of the graph into the learned node representations. Furthermore, combined with a sum readout function, these representations give rise to graph representations that can only count the number of nodes of each graph. If the readout function is the mean operator, then all graphs are embedded into the same vector. With regards to the GIN-0 and GCN models, we have bounded their Lipschitz constants with respect to the number of walks and sum of normalized walks starting from the different nodes, respectively.

To experimentally verify the above theoretical results, we trained the GIN-0 and GCN models on the IMDB-BINARY and ENZYMES graph classification datasets. For all pairs of nodes, we computed the Euclidean distance of the number of walks (resp. sum of normalized walks) of length 3 starting from them. We also computed the Euclidean distance of the representations of the nodes that emerge at the corresponding (i. e., third) layer of GIN-0 (resp. GCN). We finally computed the correlation of the two collections of Euclidean distances and the results are given in Figure 1. More details about the experimental protocol are provided in Appendix B. Clearly, the results verify our theoretical results. The distance of the number of walks is perfectly correlated with the distance of the representations generated by GIN-0 with no biases, while the distance of the sum of normalized walks is perfectly correlated with the distance of the representations produced by GCN. We also computed the Euclidean distance of the representations of the nodes that emerge at the third layer of the standard GIN-0 model (with biases), and we compared them against the distances of the number of walks. We can see that on both datasets, the emerging correlations are very high (equal to 0.99). We observed similar values of correlation on other datasets as well, which indicates that the magnitude of the bias terms of the MLPs is actually small and that our assumption of ignoring biases is by no means unrealistic.

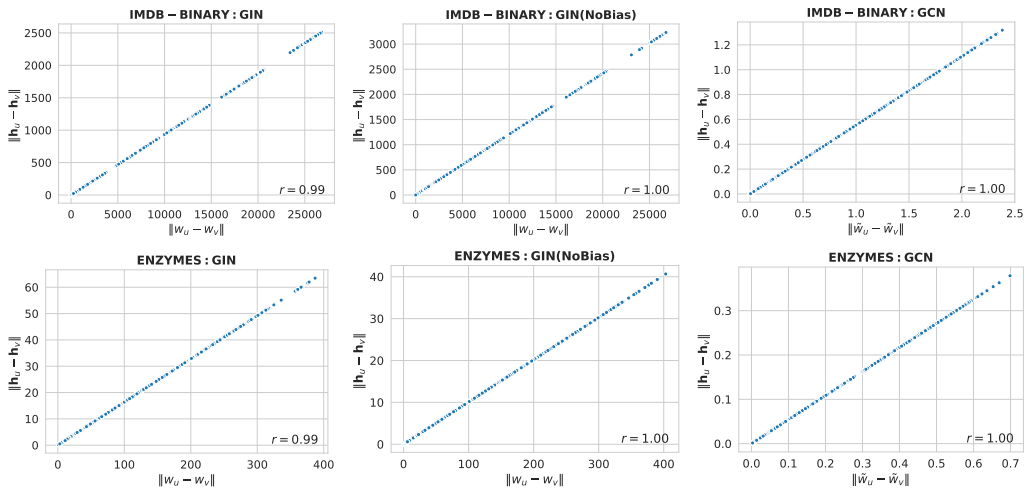

Figure 1: Euclidean distances of the representations generated at the third layer of the different models vs. Euclidean distances of the number of walks (or sum of normalized walks) of length 3 starting from the different nodes. Nodes are initially annotated with a single feature equal to 1.

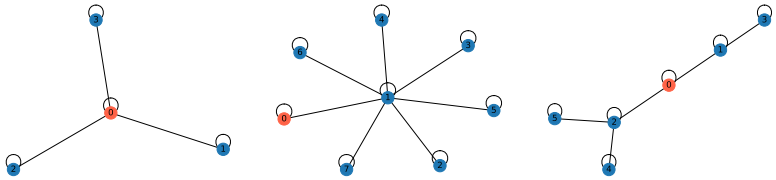

Figure 2: The number of walks of length 2 starting from the red nodes of the three graphs is equal to 10. A GIN model that consists of 2 layers embeds these three nodes close to each other (or to the same representation in case there are no biases) even though they are structurally dissimilar.

4.1 Can Structurally Dissimilar Nodes Obtain Identical Representations?

Based on the above theoretical and empirical findings, it is clear that two nodes can have dissimilar representations at the k-th layer, but obtain similar representations at the $(k+1)$-th layer of some MPNN model. For instance, for GCN, this can be the case if the two nodes have different sums of normalized walks of length k, but similar sums of normalized walks of length $k+1$. Likewise, for GIN-0, this can occur if the two nodes have different numbers of walks of length k, but similar numbers of walks of length $k+1$. But can two nodes that have different representations at the k-th layer of some MPNN model be embedded into the same vector at the $(k+1)$-th layer of the model?

Observation 1. *Let $\mathbf{h}_v^{(k)}$ denote node v's representation at the k-th layer of the GCN or the GIN-0 model (biases of MLPs are ignored). There exist nodes v and u for which $||\mathbf{h}_v^{(k)} - \mathbf{h}_u^{(k)}||_2 > 0$, but $||\mathbf{h}_v^{(k+1)} - \mathbf{h}_u^{(k+1)}||_2 = 0$ no matter what are the values of the trainable parameters of the $(k+1)$-th layer of the model.*

Figure 2 illustrates three nodes (the three red nodes) that have structurally dissimilar neighborhoods, but their representations produced by GIN-0 after two neighborhood aggregation layers are very similar to each other (or identical in case biases are omitted). Let v, u, z denote the three nodes. For all three graphs, the number of walks of length 2 starting from the red nodes is equal to 10 (i. e., $w_v^{(2)} = w_u^{(2)} = w_z^{(2)} = 10$). Note also that the three nodes have different values of degree (i. e., number of walks of length 1) from each other and thus GIN-0 could learn different representations for them after a single neighborhood aggregation layer. We also provide in Figure 3 an example of two nodes (the two red nodes) that could obtain different representations at the first layer of a GCN model, but would obtain identical representations at the second layer of the model. Let v, u denote the two nodes. For those two nodes, we have that $\tilde{w}_v^{(1)} \neq \tilde{w}_u^{(1)}$, but also that $\tilde{w}_v^{(2)} = \tilde{w}_u^{(2)} \approx 0.890$.

Figure 3: The sum of normalized walks of length 2 starting from the red nodes of the two graphs is approximately equal to 0.890. A GCN model that consists of 2 layers embeds these two nodes to the same representation even though they are structurally dissimilar.

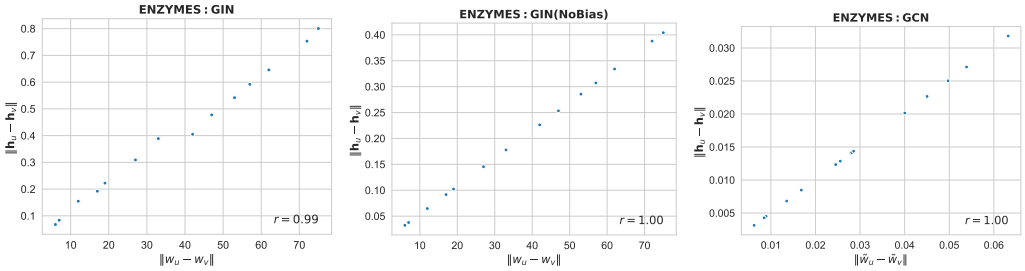

Figure 4: Euclidean distances of the representations generated at the third layer of the different models vs. Euclidean distances of the number of walks (or sum of normalized walks) of length 3 starting from the different nodes. Nodes v and u correspond to the same node in the original and the perturbed graph, respectively. Each perturbed graph has emerged by removing one node from node v's neighborhood.

In many settings, the local neighborhood of a node provides more information about its structural role than its more global neighborhood. Indeed, two nodes that have very different values of degree from each other are undoubtedly structurally very dissimilar no matter how their k-hop (for $k > 1$) neighborhoods look like. The jump connections proposed in [62] allow a model to combine the representations of the different layers (e. g., by concatenating them), thus capturing both local and more global information about each node's neighborhood.

4.2 Are MPNNs Stable under Perturbations?

We next capitalize on Theorem 1 to assess the stability of MPNNs to small perturbations such as node removal or edge removal. Note that the Lipschitz constant is a common tool to evaluate neural networks' stability to small perturbations. Our theoretical result implies that the Euclidean distance between the representation of a node and the node's new representation once some perturbation is applied depends on the decrease (or increase) in the number of walks that start from that node. We empirically validate our theoretical result on the ENZYMES dataset. We first split the dataset into training, validation and test sets, and then train the GIN-0 and GCN models on the training set. Finally, we choose one node v of some graph of the test set and use the model to produce its represention $\mathbf{h}_v$ at the third layer of each model. We then create perturbed versions of the graph. Each perturbed graph emerges from the original graph by just removing one node (and its adjacent edges) whose shortest path distance from v is at most 2. Let u denote the node of the perturbed graph that corresponds to v, and $\mathbf{h}_u$ its represention at the third layer of the two models. Figure 4 illustrates how the Euclidean distance between the two nodes varies as a function of the descrease in the number of walks. We observe that the Euclidean distance between the initial representation of the node and its new representation is highly correlated with the decrease in the number of walks due to the perturbations. Besides nodes, we also experimented with edge removal (we removed edges whose endpoints are both at distance at most 2 from v), and the results are given in Appendix C.

Note that there exist graphs where the removal of a node or of an edge can significantly degrade graph robustness. Consider for example the graph that is shown in Figure 5. The removal of node u (or of one of its adjacent edges) disconnects the graph. This will most likely have a significant impact on the representation of node v computed at the k-th layer of GIN where $k > 4$.

Indeed we observe that the number of walks of length k that start from node v decreases significantly once node u (or one of its adjacent edges) is removed. There exist 603 walks of length 6 from node v (with self-loops), but this number decreases to as few as 64 once node u is removed. We initialized 10 different GIN-0 models consisting of 6 layers and fed the graph of Figure 5 into the models (the models were not trained) along with two perturbations of the graph where either node z is removed or node u is removed (which results into a disconnected graph). We computed the norm of the difference between the representation of node v in the original graph and its representation in each of the perturbed instances at the sixth layer of the models. We found that when node z is removed, the average norm of the difference is equal to 0.0026. When node u is removed, the average norm of the difference is equal to 0.0162. This confirms our theory since by removing node u there is a large decrease in the number of walks that start from node v. On the other hand, by removing node z, the perturbation in terms of the number of walks that start from node v is not very strong, thus leading to a smaller value of the norm.

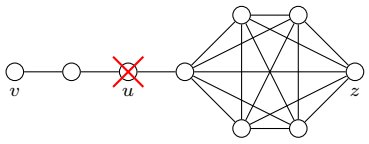

Figure 5: Example of a graph where the removal of a node disconnects the graph. This leads to a large decrease in the number of walks of length 6 that start from node v.

4.3 How Do Initial Node Features Influence Representations?

So far, we have focused on graphs which do not contain node features. For our analysis, we assumed that all nodes are annotated with the same feature(s) which is common practice in the field of graph machine learning. However, in many real-world applications (e. g., chemo-informatics), the entities that correspond to the nodes of the graph are usually associated with features and those features are not necessarily the same across all entities. Furthermore, to improve the models' expressive power, prior work typically annotates nodes with local or global features such as the degrees of the nodes or features extracted from spectral embeddings. We thus next investigate whether our previous results can be generalized to the setting where the features of a node are different from those of other nodes.

Our next result generalizes the previous results to graphs that contain node features provided that those features all point in the same direction. Formally, let $\mathcal{G} = \{G_1, \ldots, G_N\}$ be a collection of graphs, and let also $\mathcal{V} = V_1 \cup \ldots \cup V_N$ denote the set that contains the nodes of all those graphs. Then, if for any two nodes $v, u \in \mathcal{V}$, we have that $\mathbf{h}_v^{(0)}$ is a positive scalar multiple of $\mathbf{h}_u^{(0)}$, then we can still bound the Lipschitz constant of those models with respect to the weighted sum of (normalized) walks where the weights correspond to the initial node features. Let $\mathbf{w}_v^{(k)}$ denote the sum of weighted walks of length k that start from node v. The weight of a walk is equal to the feature(s) of the last visited node. For example, if there are 3 walks of length 1 that start from node v, and the features of the last visited nodes are $[2.2, 1.5]$, $[1.1, 0.75]$ and $[3.3, 2.25]$, then $\mathbf{w}_v^{(k)} = [6.6, 4.5]$. Note that the 3 vectors all point in the same direction. Let also $\tilde{\mathbf{w}}_v^{(k)}$ denote the sum of weighted normalized walks of length k where each walk $(v_1, v_2, \ldots, v_k)$ is equal to $\mathbf{h}_{v_k}^{(0)} / \left((1+d(v_2))\ldots(1+d(v_{k-1}))\sqrt{(1+d(v_1))(1+d(v_k))}\right)$.

Theorem 2. *Let $\mathcal{G} = \{G_1, \ldots, G_N\}$ be a collection of graphs. Let also $\mathcal{V} = V_1 \cup \ldots \cup V_N$ denote the set that contains the nodes of all graphs. All nodes are initially annotated with features that point in the same direction. Then, after k neighborhood aggregation layers:*

1. *GCN maps nodes to representations related to the sum of weighted normalized walks of length k starting from them:*

$$\left|\left|\mathbf{h}_v^{(k)} - \mathbf{h}_u^{(k)}\right|\right|_2 \leq \prod_{i=1}^{k} L_f^{(i)} \left|\left|\tilde{\mathbf{w}}_v^{(k)} - \tilde{\mathbf{w}}_u^{(k)}\right|\right|_2$$

   where $L_f^{(i)}$ denotes the Lipschitz constant of the fully-connected layer of the i-th neighborhood aggregation layer.

2. *Under mild assumptions (biases of MLPs are ignored), GIN-0 maps nodes to representations that capture the sum of weighted walks of length k starting from them:*

$$\left|\left|\mathbf{h}_v^{(k)} - \mathbf{h}_u^{(k)}\right|\right|_2 \leq \prod_{i=1}^{k} L_f^{(i)} \left|\left|\mathbf{w}_v^{(k)} - \mathbf{w}_u^{(k)}\right|\right|_2$$

   where $L_f^{(i)}$ denotes the Lipschitz constant of the MLP of the i-th neighborhood aggregation layer.

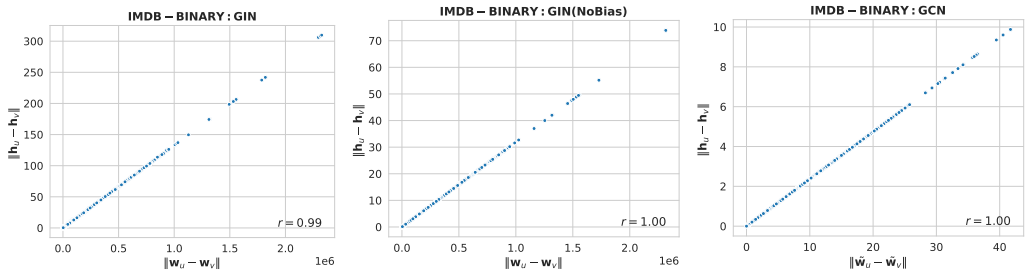

Figure 6: Euclidean distances of the representations generated at the third layer of the different models vs. Euclidean distances of the number of walks (or sum of normalized walks) of length 3 starting from the different nodes. Each node is initially annotated with a single feature equal to its degree.

To experimentally verify the above theoretical results, we annotated the nodes of all graphs of the IMDB-BINARY dataset with their degrees. Note that all node features point in the same direction. We then trained the GIN-0 and GCN models on that dataset. For pairs of nodes from graphs that belong to the test set, we computed the Euclidean distance of the sum of weighted walks (resp. sum of weighted normalized walks) of length 3 starting from them. We also computed the Euclidean distance of the representations of the nodes that emerge at the corresponding (i. e., third) layer of GIN-0 (resp. GCN). We finally computed the correlation of the two collections of Euclidean distances and the results are given in Figure 6. We observe that the empirical results verify our theoretical results. The distance of the sum of weighted walks is perfectly correlated with the distance of the representations generated by GIN-0 with no biases, while the distance of the sum of weighted normalized walks is perfectly correlated with the distance of the representations produced by GCN. We also computed the Euclidean distance of the representations of the nodes that emerge at the third layer of the standard GIN-0 model (with biases), and we compared them against the distances of the sum of weighted walks. We found that still there is almost perfect correlation between the two quantities. If the features of the nodes do not point in the same direction, then our results do not hold anymore and other techniques need to be employed to derive bounds. Deriving bounds for such kind of features is left as future work.

5 The Phenomenon of Oversquashing from Another Perspective

Our theoretical results are also related to the phenomenon of oversquashing [19, 66] which occurs in MPNNs due to large information compression through bottlenecks. Specifically, messages that are propagated from distant nodes through certain bottlenecks of the graph, turn out to have negligible impact on the root node's representation. Our theoretical results suggest that given two nodes v and u, the smaller the value of $\mathbf{w}_{vu}^{(k)}$ (where $\mathbf{w}_{vu}^{(k)}$ denotes the number of (weighted) walks of length k from node v to node u), the less the impact of the message(s) from node u to node v on node v's representation. This becomes more severe in case the norm of $\mathbf{w}_{v}^{(k)}$ is large.

To verify our claim, we constructed a graph classification task to investigate whether an MPNN model can capture the interaction between two nodes. All the generated graphs are instances of a single family of graphs. Specifically, each graph consists of two components: (1) a complete graph with n nodes; and (2) a perfectly balanced r-ary tree of height 2. The two components are connected by an edge, between one of the nodes of the complete graph and the root of the tree. We use $\text{CBT}(n, r)$ to denote such a graph with parameters n and r. Figure 7 illustrates the $\text{CBT}(4, 2)$ graph. We create a dataset that contains $\text{CBT}(n, r)$ graphs where n and r take the following values: $n \in \{4, 6, \ldots, 19\}$ and $r \in \{2, 3, \ldots, 9\}$. In fact, for each combination of n and r, we create two copies of the $\text{CBT}(n, r)$ graph. The one copy belongs to class 0 and the other copy belongs to class 1. The two graphs differ in the node feature of a single node. While all nodes of the first graph are annotated with a feature equal to 1, one of the leaves of the r-ary tree of the second graph is annotated with a feature equal to c. This gives rise to 256 graphs in

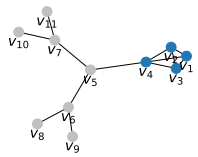

Figure 7: An example of the $\text{CBT}(4, 2)$ graph.

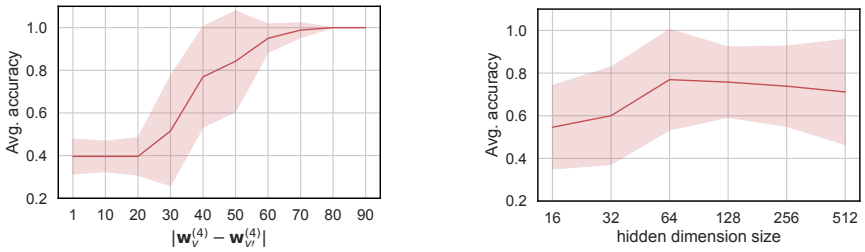

Figure 8: Left: Average accuracy on the test set of the synthetic dataset as a function of the difference of the sum of weighted walks emanating from nodes v and v'. The two nodes correspond to structurally identical nodes of the two instances of the CBT(n, r) graph. **Right**: Average accuracy on the test set of the synthetic dataset as a function of the hidden dimension size of the GIN model.

total. We split the dataset into a training, a validation and a test set with a ratio of $80\% : 10\% : 10\%$, and then train the GIN-0 model (with biases) on the first set. We set the number of neighborhood aggregation layers to 4 (i. e., shortest path distance between nodes that interact with each other). To update the node features, we use in each neighborhood aggregation layer a multi-layer perceptron that consists of 2 fully connected layers. Each fully connected layer is followed by the ReLU activation function, while batch normalization is applied to the node representations that are produced by the first fully connected layer. To make a prediction, we feed the representation of one of the nodes of the complete graph whose degree is equal to $n - 1$ to a two-layer MLP. We set the hidden dimension size to 64. We train the model for 500 epochs and use the model that achieved the lowest loss on the validation set to make predictions for the test samples. We repeat each experiment 10 times and we report the average accuracies on the test sets. Note that we set the feature c of a single leaf node to different values (e. g., 2, 11, etc.) and provide different results for each one of these values. For given n and r, let v denote one node of the complete graph (where $d(v) = n - 1$) in the first copy of the CBT(n, r) graph, and v' denote the corresponding node in the second copy. Then, we have that $|\mathbf{w}_{v'}^{(4)} - \mathbf{w}_v^{(4)}| = c$. Theorem 2 implies that the representations of nodes v and v' of the two copies (that belong to different classes) will be close to each other in case c is small.

The results are provided in Figure 8 (Left). We observe that when $|\mathbf{w}_{v'}^{(4)} - \mathbf{w}_v^{(4)}|$ is small, the model achieves very low levels of performance. This is not surprising since for each combination of n and r, there are two nodes (v and v') that belong to different classes, but their representations at the fourth layer of the model are very similar to each other. It is very hard then for the MLP that produces the output to distinguish between samples of class 0 and samples of class 1. The model's performance increases as $|\mathbf{w}_{v'}^{(4)} - \mathbf{w}_v^{(4)}|$ increases. For $|\mathbf{w}_{v'}^{(4)} - \mathbf{w}_v^{(4)}| \geq 80$, the model correctly classifies all test samples. Our results provide a different perspective for the phenomenon of oversquashing. They indicate that it predominantly arises when the number of walks from one node to some other node is disproportionally small compared to the total walks originating from the former. We also investigate what is the impact of the model's hidden dimension size on its ability to capture the dependence between the two nodes. We set $c = 41$ (i. e., $|\mathbf{w}_{v'}^{(4)} - \mathbf{w}_v^{(4)}| = 40$), and we compute the model's average accuracy on the test sets as a function of the hidden dimension size. The results are shown in Figure 8 (Right). While for small values, it appears that the increase in the hidden dimension size also improves the model's performance, there is no further increase in performance for hidden dimension sizes beyond 64. Our results thus indicate that this limitation of MPNNs cannot be addressed just by increasing the hidden dimension size and that other approaches need to be employed.

6 Conclusion

In this paper, we focused on four well-established MPNN models and investigated what properties of graphs these models can capture. First, we considered the case where no node features are available. We found that two models capture no structural properties of graphs, while the rest of the models learn node representations that capture the sum of (normalized) walks emanating from the different nodes. We established Lipschitz bounds for these models with respect to the sum of (normalized) walks. We generalized the above results in case of node features that point in the same direction. Finally, we provided a different perspective for the phenomenon of oversquashing.

References

[1] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020. 1

[2] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010. 1

[3] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs. *ACM Computing Surveys*, 54(4):1–37, 2021. 1

[4] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020. 1

[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272, 2017. 1, 2, 3

[6] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations*, 2019. 1, 2, 3, 17

[7] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 4602–4609, 2019. 1, 2

[8] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 2021. 2

[9] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized Kernels Between Labeled Graphs. In *Proceedings of the 20th International Conference on Machine Learning*, pages 321–328, 2003. 2

[10] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. Springer, 2003. 2

[11] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 74–81, 2005. 2

[12] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 488–495, 2009. 2

[13] Nils Kriege and Petra Mutzel. Subgraph Matching Kernels for Attributed Graphs. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 291–298, 2012. 2

[14] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021. 2

[15] Yujia Huang, Huan Zhang, Yuanyuan Shi, J Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. In *Advances in Neural Information Processing Systems*, pages 22745–22757, 2021. 2

[16] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. In *6th International Conference on Learning Representations*, 2018. 2, 3, 14

[17] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An End-to-End Deep Learning Architecture for Graph Classification. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4438–4445, 2018. 2, 3, 13

[18] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations*, 2017. 2, 3, 14

[19] Uri Alon and Eran Yahav. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *9th International Conference on Learning Representations*, 2021. 2, 8

[20] Alessandro Sperduti and Antonina Starita. Supervised Neural Networks for the Classification of Structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997. 2

[21] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[22] Alessio Micheli. Neural Network for Graphs: A Contextual Constructive Approachs. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009. 2

[23] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally connected networks on Graphs. In *2nd International Conference on Learning Representations*, 2014. 2

[24] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[25] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[26] James Atwood and Don Towsley. Diffusion-Convolutional Neural Networks . In *Advances in Neural Information Processing Systems*, volume 29, pages 1993–2001, 2016.

[27] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016. 2

[28] Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational Pooling for Graph Representations. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4663–4673, 2019. 2

[29] Younjoo Seo, Andreas Loukas, and Nathanaël Perraudin. Discriminative structural graph classification. *arXiv preprint arXiv:1905.13422*, 2019.

[30] David Buterez, Jon Paul Janet, Steven J Kiddle, Dino Oglic, and Pietro Liò. Graph neural networks with adaptive readouts. *arXiv preprint arXiv:2211.04952*, 2022.

[31] Michail Chatzianastasis, Johannes Lutzeyer, George Dasoulas, and Michalis Vazirgiannis. Graph ordering attention networks. In *Proceedings of the 37the AAAI Conference on Artificial Intelligence*, pages 7006–7014, 2023. 2

[32] Yilun Jin, Guojie Song, and Chuan Shi. GraLSP: Graph Neural Networks with Local Structural Patterns. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 4361–4368, 2020. 2

[33] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *Proceedings of the 36th International Conference on Machine Learning*, pages 21–29, 2019. 2

[34] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic Graph Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 4868–4879, 2019. 2

[35] Claudio Gallicchio and Alessio Micheli. Fast and Deep Graph Neural Networks. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 3898–3905, 2020. 2

[36] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems*, volume 32, pages 4800–4810, 2018. 2

[37] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2083–2092, 2019. 2

[38] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning Convolutional Neural Networks for Graphs. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2014–2023, 2016. 2

[39] Giannis Nikolentzos and Michalis Vazirgiannis. Random Walk Graph Neural Networks. In *Advances in Neural Information Processing Systems*, pages 16211–16222, 2020.

[40] Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. Permute Me Softly: Learning Soft Permutations for Graph Representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):5087–5098, 2023. 2

[41] Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *Transactions on Machine Learning Research*, 2023. 2

[42] Yuanqing Wang and Kyunghyun Cho. Non-convolutional graph neural networks. In *Advances in Neural Information Processing Systems*, 2024. 2

[43] Sandra Kiefer. The Weisfeiler-Leman Algorithm: An Exploration of its Power. *ACM SIGLOG News*, 7(3):5–27, 2020. 2

[44] Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. In *Advances in Neural Information Processing Systems*, volume 34, 2020. 2

[45] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and Equivariant Graph Networks. In *7th International Conference on Learning Representations*, 2019. 2

[46] Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. *Advances in Neural Information Processing Systems*, 33:14143–14155, 2020. 2

[47] George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring Graph Neural Networks for Node Disambiguation. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 2126–2132, 2020. 2

[48] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random Features Strengthen Graph Neural Networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining*, pages 333–341, 2021. 2

[49] Ralph Abboud, Ismail Ilkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. *arXiv preprint arXiv:2010.01179*, 2020. 2

[50] Jiaxuan You, Jonathan M Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pages 10737–10745, 2021. 2

[51] Muhammet Balcilar, Pierre Héroux, Benoit Gauzere, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning*, pages 599–608. PMLR, 2021. 2

[52] Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pages 1026–1037. PMLR, 2021. 2

[53] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and Lehman Go Cellular: CW Networks. In *Advances in Neural Information Processing Systems*, pages 2625–2640, 2021. 2

[54] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In *International Conference on Machine Learning*, pages 748–758. PMLR, 2021. 2

[55] Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop graph neural networks. *Neural Networks*, 130:195–205, 2020. 2

[56] Muhan Zhang and Pan Li. Nested Graph Neural Networks. In *Advances in Neural Information Processing Systems*, pages 15734–15747, 2021.

[57] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. *arXiv preprint arXiv:2110.02910*, 2021. 2

[58] Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. *Advances in Neural Information Processing Systems*, 34:1713–1726, 2021. 3

[59] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021. 3

[60] Giannis Nikolentzos, Michail Chatzianastasis, and Michalis Vazirgiannis. Weisfeiler and Leman go Hyperbolic: Learning Distance Preserving Node Representations. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, pages 1037–1054, 2023. 3

[61] Ching-Yao Chuang and Stefanie Jegelka. Tree Mover's Distance: Bridging Graph Metrics and Stability of Graph Neural Networks. In *Advances in Neural Information Processing Systems*, 2022. 3

[62] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation Learning on Graphs with Jumping Knowledge Networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5453–5462, 2018. 3, 6

[63] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From Local Structures to Size Generalization in Graph Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning*, pages 11975–11986, 2021. 3

[64] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2022. 4

[65] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking Graph Neural Networks. *arXiv preprint arXiv:2003.00982*, 2020. 4

[66] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *9th International Conference on Learning Representations*, 2021. 8

A Proof of Theorem 1

We assume that all the nodes of the graph are initially annotated with a single feature equal to 1.

A.1 DGCNN

The DGCNN model updates node representations as follows [17]:

$$\mathbf{h}_v^{(k)} = f\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{d(v)+1} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right)$$

We will show by induction that the model maps all nodes to the same vector representation. We first assume that $\mathbf{h}_v^{(k-1)} = \mathbf{h}_u^{(k-1)} = \mathbf{h}^{(k-1)}$ for all $v, u \in V$. This is actually true for $k = 1$ since $\mathbf{h}_v^{(0)} = 1$ for all $v \in V$. Then, we have that $\mathbf{W}^{(k)} \mathbf{h}_v^{(k-1)} = \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)} = \mathbf{W}^{(k)} \mathbf{h}^{(k-1)}$ for all $v, u \in V$. We also have that:

$$\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{d(v)+1} = 1$$

Thus, we finally have that:

$$\mathbf{h}_v^{(k)} = f\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{d(v)+1} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right) = f\left(\mathbf{W}^{(k)} \mathbf{h}^{(k-1)}\right)$$

for all $v \in V$. We have shown that this variant of the GCN model produces the same representation for all nodes of all graphs and thus, it cannot capture any structural information about the neighborhood of each node.

A.2 GAT

The GAT model updates node representations as follows [16]:

$$\mathbf{h}_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right)$$

where α_{vu} is an attention coefficient that indicates the importance of node u's features to node v. Once again, we assume that all the nodes of the graph are initially annotated with a single feature equal to 1. We will show by induction that the model maps all nodes to the same vector representation. We first assume that $\mathbf{h}_v^{(k-1)} = \mathbf{h}_u^{(k-1)} = \mathbf{h}^{(k-1)}$ for all $v, u \in V$. This is actually true for $k = 1$ since $\mathbf{h}_v^{(0)} = 1$ for all $v \in V$. Then, we have that $\mathbf{W}^{(k)} \mathbf{h}_v^{(k-1)} = \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)} = \mathbf{W}^{(k)} \mathbf{h}^{(k-1)}$ for all $v, u \in V$. Since the attention coefficients are normalized, we have:

$$\sum_{u \in \mathcal{N}(v)} \alpha_{vu} = 1$$

Thus, we finally have that:

$$\mathbf{h}_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}\right) = \sigma\left(\mathbf{W}^{(k)} \mathbf{h}^{(k-1)}\right)$$

for all $v \in V$. We have shown that the GAT model produces the same representation for all nodes of all graphs and thus, it cannot capture any structural information about the neighborhood of each node.

A.3 GCN

The GCN model updates node representations as follows [18]:

$$\mathbf{h}_v^{(k)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{\mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}}{\sqrt{(1 + d(v))(1 + d(u))}}\right)$$

Note that the GCN model (as decribed in the original paper [18]) does not perform an affine transformation of the node features, but instead a linear transformation. In other words, no biases are present. Thus, the following holds:

$$\begin{aligned}
\mathbf{h}_v^{(k)} &= \text{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{\mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)}}{\sqrt{(1 + d(v))(1 + d(u))}}\right) \\
&= \text{ReLU}\left(\mathbf{W}^{(k)} \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{\mathbf{h}_u^{(k-1)}}{\sqrt{(1 + d(v))(1 + d(u))}}\right)
\end{aligned}$$

We next prove the following Lemma which will be useful in our analysis.

Lemma 1. *Let $\mathbf{W}$ denote a matrix. Let also $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ denote a set of vectors such that given any two vectors from the set, one is a positive scalar multiple of the other, i. e., if $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, then $\mathbf{x}_i = a\mathbf{x}_j$ where $a > 0$. Let also $c_i > 0$ for all $i \in \{1, \ldots, M\}$. Then, the following holds:*

$$\sum_{i=1}^{M} c_i \mathit{ReLU}(\mathbf{W}\, \mathbf{x}_i) = \mathit{ReLU}\left(\mathbf{W} \sum_{i=1}^{M} c_i \mathbf{x}_i\right)$$

Proof. For any scalar a, we have that $\mathbf{W}(a\mathbf{x}) = a\mathbf{W}\mathbf{x}$. Furthermore, for $a > 0$, we have that $\text{ReLU}(\mathbf{W}(a\mathbf{x})) = \text{ReLU}(a\mathbf{W}\mathbf{x}) = a\text{ReLU}(\mathbf{W}\mathbf{x})$. Suppose that $\mathbf{x}_2 = a_2\mathbf{x}_1$, $\mathbf{x}_3 = a_3\mathbf{x}_1, \ldots,$

$\mathbf{x}_M = a_M \mathbf{x}_1$. Then, we have that:

$$\begin{aligned}
\sum_{i=1}^{M} c_i \text{ReLU}(\mathbf{W}\,\mathbf{x}_i) &= c_1 \text{ReLU}(\mathbf{W}\,\mathbf{x}_1) + \sum_{i=2}^{M} c_i \text{ReLU}(\mathbf{W}\,a_i\mathbf{x}_1) \\
&= c_1 \text{ReLU}(\mathbf{W}\,\mathbf{x}_1) + \sum_{i=2}^{M} c_i a_i \text{ReLU}(\mathbf{W}\,\mathbf{x}_1) \\
&= (c_1 + a_2 c_2 + \ldots + a_M c_M)\text{ReLU}(\mathbf{W}\,\mathbf{x}_1) \\
&= \text{ReLU}\Big((c_1 + a_2 c_2 + \ldots + a_M c_M)\mathbf{W}\,\mathbf{x}_1\Big) \\
&= \text{ReLU}\Big(\mathbf{W}(c_1 + a_2 c_2 + \ldots + a_M c_M)\mathbf{x}_1\Big) \\
&= \text{ReLU}\Big(\mathbf{W}\sum_{i=1}^{M} c_i \mathbf{x}_i\Big)
\end{aligned}$$

which concludes the proof. □

We also prove the following Lemma.

Lemma 2. *Let $\mathcal{V}$ denote the set of nodes of all graphs and let $\mathcal{X}^{(k-1)} = \{\!\{\mathbf{h}_1^{(k-1)}, \ldots, \mathbf{h}_{|\mathcal{V}|}^{(k-1)}\}\!\}$ be the multiset of node representations that emerged at the $(k-1)$-th layer of the GCN model. Suppose that given any two vectors from this multiset, one is a scalar multiple of the other, i. e., if $\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)} \in \mathcal{X}^{(k-1)}$, then $\mathbf{h}_i^{(k-1)} = a\mathbf{h}_j^{(k-1)}$ where $a > 0$. Then, the same holds for the node representations that emerge at the k-th layer of the model, i. e., for any two vectors $\mathbf{h}_i^{(k)}, \mathbf{h}_j^{(k)} \in \mathcal{X}^{(k)} = \{\!\{\mathbf{h}_1^{(k)}, \ldots, \mathbf{h}_{|\mathcal{V}|}^{(k)}\}\!\}$, we have that $\mathbf{h}_i^{(k)} = a\mathbf{h}_j^{(k)}$ where $a > 0$.*

Proof. For $a > 0$, we have that $\mathbf{W}(a\mathbf{x}) = a\mathbf{W}\mathbf{x}$. Furthermore, we also have that $\text{ReLU}(a\mathbf{x}) = a\text{ReLU}(\mathbf{x})$. We have assumed that given an arbitrary node $w \in \mathcal{V}$, then for any node $u \in \mathcal{V}$, we have that $\mathbf{h}_u^{(k-1)} = a_u \mathbf{h}_w^{(k-1)}$. Then, given any node $v \in \mathcal{V}$, its representation is updated as follows:

$$\begin{aligned}
\mathbf{h}_v^{(k)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v)\cup\{v\}} \frac{\mathbf{W}^{(k)}\mathbf{h}_u^{(k-1)}}{\sqrt{(1+d(v))(1+d(u))}}\right) &= \text{ReLU}\left(\sum_{u \in \mathcal{N}(v)\cup\{v\}} \frac{\mathbf{W}^{(k)} a_u \mathbf{h}_w^{(k-1)}}{\sqrt{(1+d(v))(1+d(u))}}\right) \\
&= \sum_{u \in \mathcal{N}(v)\cup\{v\}} a_u \text{ReLU}\left(\frac{\mathbf{W}^{(k)}\mathbf{h}_w^{(k-1)}}{\sqrt{(1+d(v))(1+d(u))}}\right) \\
&= \text{ReLU}\left(\frac{\mathbf{W}^{(k)}\mathbf{h}_w^{(k-1)}}{\sqrt{(1+d(v))(1+d(u))}}\right)
\end{aligned}$$

where $c = \sum_{u \in \mathcal{N}(v)\cup\{v\}} a_u > 0$. Since node v is an arbitrary node, we observe that the representations of all nodes point in the same direction. □

Thus, the above Lemma suggests that the node representations produced by GCN are either scalar multiples of each other and they point in the same direction or are all equal to the all-zeros vector.

Let $L_f^{(k)}$ denote the Lipschitz constant associated with the fully connected layer of the k-th neighborhood aggregation layer of GCN. In what follows, we also denote $\mathcal{N}(v)\cup\{v\}$ as $\mathcal{N}^+(v)$ and $d(v)+1$ as d_v^+. Then, we have:

$$\begin{aligned}
||\mathbf{h}_v^{(1)} - \mathbf{h}_u^{(1)}||_2 &= \left|\left|\text{ReLU}\left(\sum_{w \in \mathcal{N}^+(v)} \frac{\mathbf{W}^{(1)}\mathbf{h}_w^{(0)}}{\sqrt{d_v^+ d_w^+}}\right) - \text{ReLU}\left(\sum_{w \in \mathcal{N}^+(u)} \frac{\mathbf{W}^{(1)}\mathbf{h}_w^{(0)}}{\sqrt{d_u^+ d_w^+}}\right)\right|\right|_2 \\
&= \left|\left|\text{ReLU}\left(\mathbf{W}^{(1)}\sum_{w \in \mathcal{N}^+(v)} \frac{\mathbf{h}_w^{(0)}}{\sqrt{d_v^+ d_w^+}}\right) - \text{ReLU}\left(\mathbf{W}^{(1)}\sum_{w \in \mathcal{N}^+(u)} \frac{\mathbf{h}_w^{(0)}}{\sqrt{d_u^+ d_w^+}}\right)\right|\right|_2
\end{aligned}$$

$$\leq L_f^{(1)} \left|\left| \sum_{w \in \mathcal{N}^+(v)} \frac{1}{\sqrt{d_v^+ d_w^+}} \quad - \sum_{w \in \mathcal{N}^+(u)} \frac{1}{\sqrt{d_u^+ d_w^+}} \right|\right|_2$$

$$||\mathbf{h}_v^{(2)} - \mathbf{h}_u^{(2)}||_2 = \left|\left| \text{ReLU}\left(\sum_{w \in \mathcal{N}^+(v)} \frac{\mathbf{W}^{(2)} \mathbf{h}_w^{(1)}}{\sqrt{d_v^+ d_w^+}} \right) - \text{ReLU}\left(\sum_{w \in \mathcal{N}^+(u)} \frac{\mathbf{W}^{(2)} \mathbf{h}_w^{(1)}}{\sqrt{d_u^+ d_w^+}} \right) \right|\right|_2$$

$$= \left|\left| \text{ReLU}\left(\mathbf{W}^{(2)} \sum_{w \in \mathcal{N}^+(v)} \frac{\mathbf{h}_w^{(1)}}{\sqrt{d_v^+ d_w^+}} \right) - \text{ReLU}\left(\mathbf{W}^{(2)} \sum_{w \in \mathcal{N}^+(u)} \frac{\mathbf{h}_w^{(1)}}{\sqrt{d_u^+ d_w^+}} \right) \right|\right|_2$$

$$\leq L_f^{(2)} \left|\left| \sum_{w \in \mathcal{N}^+(v)} \frac{1}{\sqrt{d_v^+ d_w^+}} \text{ReLU}\left(\sum_{x \in \mathcal{N}^+(w)} \frac{\mathbf{W}^{(1)} \mathbf{h}_w^{(0)}}{\sqrt{d_w^+ d_x^+}} \right) - \sum_{w \in \mathcal{N}^+(u)} \frac{1}{\sqrt{d_u^+ d_w^+}} \text{ReLU}\left(\sum_{x \in \mathcal{N}^+(w)} \frac{\mathbf{W}^{(1)} \mathbf{h}_w^{(0)}}{\sqrt{d_w^+ d_x^+}} \right) \right|\right|_2$$

$$= L_f^{(2)} \left|\left| \text{ReLU}\left(\mathbf{W}^{(1)} \sum_{w \in \mathcal{N}^+(v)} \sum_{x \in \mathcal{N}^+(w)} \frac{\mathbf{h}_w^{(0)}}{\sqrt{d_v^+ d_w^+} \sqrt{d_w^+ d_x^+}} \right) - \text{ReLU}\left(\mathbf{W}^{(1)} \sum_{w \in \mathcal{N}^+(u)} \sum_{x \in \mathcal{N}^+(w)} \frac{\mathbf{h}_w^{(0)}}{\sqrt{d_u^+ d_w^+} \sqrt{d_w^+ d_x^+}} \right) \right|\right|_2$$

(due to Lemmas 1, 2)

$$\leq L_f^{(2)} L_f^{(1)} \left|\left| \sum_{w \in \mathcal{N}^+(v)} \sum_{x \in \mathcal{N}^+(w)} \frac{1}{\sqrt{d_v^+ d_w^+} \sqrt{d_w^+ d_x^+}} - \sum_{w \in \mathcal{N}^+(u)} \sum_{x \in \mathcal{N}^+(w)} \frac{1}{\sqrt{d_u^+ d_w^+} \sqrt{d_w^+ d_x^+}} \right|\right|_2$$

$$= L_f^{(2)} L_f^{(1)} \left|\left| \sum_{w \in \mathcal{N}^+(v)} \sum_{x \in \mathcal{N}^+(w)} \frac{1}{d_w^+ \sqrt{d_v^+ d_x^+}} - \sum_{w \in \mathcal{N}^+(u)} \sum_{x \in \mathcal{N}^+(w)} \frac{1}{d_w^+ \sqrt{d_u^+ d_x^+}} \right|\right|_2$$

$$\vdots$$

$$||\mathbf{h}_v^{(K)} - \mathbf{h}_u^{(K)}||_2 = \left|\left| \text{ReLU}\left(\sum_{w \in \mathcal{N}^+(v)} \frac{\mathbf{W}^{(K)} \mathbf{h}_w^{(K-1)}}{\sqrt{d_v^+ d_w^+}} \right) - \text{ReLU}\left(\sum_{w \in \mathcal{N}^+(u)} \frac{\mathbf{W}^{(K)} \mathbf{h}_w^{(K-1)}}{\sqrt{d_u^+ d_w^+}} \right) \right|\right|_2$$

$$= \left|\left| \text{ReLU}\left(\mathbf{W}^{(K)} \sum_{w \in \mathcal{N}^+(v)} \frac{\mathbf{h}_w^{(K-1)}}{\sqrt{d_v^+ d_w^+}} \right) - \text{ReLU}\left(\mathbf{W}^{(K)} \sum_{w \in \mathcal{N}^+(u)} \frac{\mathbf{h}_w^{(K-1)}}{\sqrt{d_u^+ d_w^+}} \right) \right|\right|_2$$

$$\leq L_f^{(K)} \left|\left| \sum_{w \in \mathcal{N}^+(v)} \frac{1}{\sqrt{d_v^+ d_w^+}} \text{ReLU}\left(\sum_{x \in \mathcal{N}^+(w)} \frac{\mathbf{W}^{(K-1)} \mathbf{h}_w^{(K-2)}}{\sqrt{d_w^+ d_x^+}} \right) \right.$$

$$\left. - \sum_{w \in \mathcal{N}^+(u)} \frac{1}{\sqrt{d_u^+ d_w^+}} \text{ReLU}\left(\sum_{x \in \mathcal{N}^+(w)} \frac{\mathbf{W}^{(K-1)} \mathbf{h}_w^{(K-2)}}{\sqrt{d_w^+ d_x^+}} \right) \right|\right|_2$$

$$= L_f^{(K)} \left|\left| \text{ReLU}\left(\mathbf{W}^{(K-1)} \sum_{w \in \mathcal{N}^+(v)} \sum_{x \in \mathcal{N}^+(w)} \frac{\mathbf{h}_w^{(K-2)}}{\sqrt{d_v^+ d_w^+} \sqrt{d_w^+ d_x^+}} \right) \right.$$

$$\left. - \text{ReLU}\left(\mathbf{W}^{(K-1)} \sum_{w \in \mathcal{N}^+(u)} \sum_{x \in \mathcal{N}^+(w)} \frac{\mathbf{h}_w^{(K-2)}}{\sqrt{d_u^+ d_w^+} \sqrt{d_w^+ d_x^+}} \right) \right|\right|_2 \text{(due to Lemmas 1, 2)}$$

$$\leq L_f^{(K)} L_f^{(K-1)} \dots L_f^{(2)} \left|\left| \sum_{w \in \mathcal{N}^+(v)} \dots \sum_{x \in \mathcal{N}^+(y)} \frac{1}{\sqrt{d_v^+ d_w^+} \dots \sqrt{d_x^+ d_y^+}} \text{ReLU}\left(\sum_{z \in \mathcal{N}^+(x)} \frac{\mathbf{W}^{(1)} \mathbf{h}_z^{(0)}}{\sqrt{d_x^+ d_z^+}} \right) \right|\right|_2$$

$$- \sum_{w \in \mathcal{N}^+(u)} \dots \sum_{x \in \mathcal{N}^+(y)} \frac{1}{\sqrt{d_u^+ d_w^+} \dots \sqrt{d_y^+ d_x^+}} \text{ReLU}\left(\sum_{z \in \mathcal{N}^+(x)} \frac{\mathbf{W}^{(1)} \mathbf{h}_z^{(0)}}{\sqrt{d_x^+ d_z^+}} \right) \Bigg\|_2$$

$$= L_f^{(K)} L_f^{(K-1)} \dots L_f^{(2)} \Bigg\| \text{ReLU}\left(\mathbf{W}^{(1)} \sum_{w \in \mathcal{N}^+(v)} \dots \sum_{x \in \mathcal{N}^+(y)} \sum_{z \in \mathcal{N}^+(x)} \frac{\mathbf{h}_z^{(0)}}{\sqrt{d_v^+ d_w^+} \dots \sqrt{d_x^+ d_y^+} \sqrt{d_x^+ d_z^+}} \right)$$

$$- \text{ReLU}\left(\mathbf{W}^{(1)} \sum_{w \in \mathcal{N}^+(u)} \sum_{x \in \mathcal{N}^+(y)} \sum_{z \in \mathcal{N}^+(x)} \frac{\mathbf{h}_z^{(0)}}{\sqrt{d_u^+ d_w^+} \dots \sqrt{d_y^+ d_x^+} \sqrt{d_x^+ d_z^+}} \right) \Bigg\|_2 \text{ (due to Lemmas 1, 2)}$$

$$\leq L_f^{(K)} L_f^{(K-1)} \dots L_f^{(1)} \Bigg\| \sum_{w \in \mathcal{N}^+(v)} \dots \sum_{x \in \mathcal{N}^+(y)} \sum_{z \in \mathcal{N}^+(x)} \frac{1}{\sqrt{d_v^+ d_w^+} \dots \sqrt{d_y^+ d_x^+} \sqrt{d_x^+ d_z^+}}$$

$$- \sum_{w \in \mathcal{N}^+(u)} \dots \sum_{x \in \mathcal{N}^+(y)} \sum_{z \in \mathcal{N}^+(x)} \frac{1}{\sqrt{d_u^+ d_w^+} \dots \sqrt{d_y^+ d_x^+} \sqrt{d_x^+ d_z^+}} \Bigg\|_2$$

$$= L_f^{(K)} L_f^{(K-1)} \dots L_f^{(1)} \Bigg\| \sum_{w \in \mathcal{N}^+(v)} \dots \sum_{x \in \mathcal{N}^+(y)} \sum_{z \in \mathcal{N}^+(x)} \frac{1}{d_w^+ \dots d_y^+ d_x^+ \sqrt{d_v^+ d_z^+}}$$

$$- \sum_{w \in \mathcal{N}^+(u)} \dots \sum_{x \in \mathcal{N}^+(y)} \sum_{z \in \mathcal{N}^+(x)} \frac{1}{d_w^+ \dots d_y^+ d_x^+ \sqrt{d_u^+ d_z^+}} \Bigg\|_2$$

It turns out that the node representations learned at the k-th layer of a GCN model are related to the sum of normalized walks of length k starting from each node. Given a walk of length k consisting of the following nodes $(v_1, v_2, \dots, v_k)$, the walk is normalized by the product of the degrees of nodes $v_2, \dots, v_{k-1}$ each increased by 1 and of the square roots of the degrees of nodes v_1 and v_k also increased by 1. Thus, the contribution of each walk is inversely proportional to the degrees of the nodes of which the walk is composed.

A.4 GIN-0

The GIN-0 model updates node representations as follows [6]:

$$\mathbf{h}_v^{(k)} = \text{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k-1)} \right) = \text{MLP}^{(k)}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \mathbf{h}_u^{(k-1)} \right) \quad (1)$$

We make the following assumption.

Assumption 1. We assume that the biases of the fully-connected layers of all MLPs are equal to zero vectors.

Our results are also valid when given fully-connected layers of the form $\text{fc}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$, the following holds $||\mathbf{b}|| \ll ||\mathbf{W}\mathbf{x}||$. Note that the activation function of the MLPs of the GIN-0 model is the ReLU function [6]. Without loss of generality, we assume that the MLPs consist of two fully-connected layers. Given the above, the update function of the GIN-0 model takes the following form:

$$\text{MLP}^{(k)}(\mathbf{x}) = \text{ReLU}\left(\mathbf{W}_2^{(k)} \text{ReLU}\left(\mathbf{W}_1^{(k)} \mathbf{x} \right) \right) \quad (2)$$

We next prove the following Lemma which will be useful in our analysis.

Lemma 3. *Let MLP denote the model defined in Equation (2) above. Let also $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$ denote a set of vectors such that given any two vectors from the set, one is a scalar multiple of the other, i. e., if $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, then $\mathbf{x}_i = a\mathbf{x}_j$ where $a > 0$. Then, the following holds:*

$$\sum_{i=1}^{M} \textit{MLP}(\mathbf{x}_i) = \textit{MLP}\left(\sum_{i=1}^{M} \mathbf{x}_i \right)$$

Proof. For $a > 0$, we have that $\mathbf{W}(a\mathbf{x}) = a\mathbf{W}\mathbf{x}$. Furthermore, we also have that $\text{ReLU}(a\mathbf{x}) = a\text{ReLU}(\mathbf{x})$. Then, we have:

$$\text{MLP}(a\mathbf{x}) = \text{ReLU}\Bigg(\mathbf{W}_2\,\text{ReLU}\Bigg(\mathbf{W}_1(a\mathbf{x})\Bigg)\Bigg) = a\text{ReLU}\Bigg(\mathbf{W}_2\,\text{ReLU}\Bigg(\mathbf{W}_1(\mathbf{x})\Bigg)\Bigg) = a\text{MLP}(\mathbf{x})$$

Suppose that $\mathbf{x}_2 = a_2\mathbf{x}_1$, $\mathbf{x}_3 = a_3\mathbf{x}_1, \ldots, \mathbf{x}_M = a_M\mathbf{x}_1$. Then, we have that:

$$\begin{aligned}
\sum_{i=1}^{M}\text{MLP}(\mathbf{x}_i) &= \text{MLP}(\mathbf{x}_1) + \sum_{i=2}^{M}\text{MLP}(a_i\mathbf{x}_1) \\
&= \text{MLP}(\mathbf{x}_1) + \sum_{i=2}^{M} a_i\text{MLP}(\mathbf{x}_1) \\
&= (1 + a_2 + \ldots + a_M)\text{MLP}(\mathbf{x}_1) \\
&= \text{MLP}\Big((1 + a_2 + \ldots + a_M)\mathbf{x}_1\Big) \\
&= \text{MLP}\Bigg(\sum_{i=1}^{M}\mathbf{x}_i\Bigg)
\end{aligned}$$

□

It is trivial to generalize the above Lemma to MLPs that contain more than two layers. We also prove the following Lemma.

Lemma 4. *Let the MLPs of the GIN-0 model be instances of the MLP of Equation (2) above. Let $\mathcal{V}$ denote the set of nodes of all graphs and let $\mathcal{X}^{(k-1)} = \{\!\{\mathbf{h}_1^{(k-1)}, \ldots, \mathbf{h}_{|\mathcal{V}|}^{(k-1)}\}\!\}$ be the multiset of node representations that emerged at the $(k-1)$-th layer of the model. Suppose that given any two vectors from this multiset, one is a scalar multiple of the other, i. e., if $\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)} \in \mathcal{X}^{(k-1)}$, then $\mathbf{h}_i^{(k-1)} = a\mathbf{h}_j^{(k-1)}$ where $a > 0$. Then, the same holds for the node representations that emerge at the k-th layer of the model, i. e., for any two vectors $\mathbf{h}_i^{(k)}, \mathbf{h}_j^{(k)} \in \mathcal{X}^{(k)} = \{\!\{\mathbf{h}_1^{(k)}, \ldots, \mathbf{h}_{|\mathcal{V}|}^{(k)}\}\!\}$, we have that $\mathbf{h}_i^{(k)} = a\mathbf{h}_j^{(k)}$ where $a > 0$.*

Proof. For $a > 0$, we have that $\mathbf{W}(a\mathbf{x}) = a\mathbf{W}\mathbf{x}$. Furthermore, we also have that $\text{ReLU}(a\mathbf{x}) = a\text{ReLU}(\mathbf{x})$. Then, we have:

$$\text{MLP}(a\mathbf{x}) = \text{ReLU}\Bigg(\mathbf{W}_2\,\text{ReLU}\Bigg(\mathbf{W}_1(a\mathbf{x})\Bigg)\Bigg) = a\text{ReLU}\Bigg(\mathbf{W}_2\,\text{ReLU}\Bigg(\mathbf{W}_1(\mathbf{x})\Bigg)\Bigg) = a\text{MLP}(\mathbf{x})$$

We have assumed that given an arbitrary node $w \in \mathcal{V}$, then for any node $u \in \mathcal{V}$, we have that $\mathbf{h}_u^{(k-1)} = a_u\mathbf{h}_w^{(k-1)}$. Then, given any node $v \in \mathcal{V}$, its representation is updated as follows:

$$\begin{aligned}
\mathbf{h}_v^{(k)} = \text{MLP}^{(k)}\Bigg(\sum_{u\in\mathcal{N}(v)\cup\{v\}}\mathbf{h}_u^{(k-1)}\Bigg) &= \text{MLP}^{(k)}\Bigg(\sum_{u\in\mathcal{N}(v)\cup\{v\}}a_u\mathbf{h}_w^{(k-1)}\Bigg) \\
&= \sum_{u\in\mathcal{N}(v)\cup\{v\}}a_u\text{MLP}^{(k)}\Bigg(\mathbf{h}_w^{(k-1)}\Bigg) \\
&= c\text{MLP}^{(k)}\Bigg(\mathbf{h}_w^{(k-1)}\Bigg)
\end{aligned}$$

where $c = \sum_{u\in\mathcal{N}(v)\cup\{v\}} a_u > 0$. Since node v is an arbitrary node, we can conclude that all nodes obtain representations that are scalar multiples of each other and they point in the same direction. □

Thus, the above Lemma suggests that for MLPs of the form of Equation (2), the node representations produced by GIN-0 are either scalar multiples of each other and they point in the same direction or are all equal to the all-zeros vector.

Let $L_f^{(k)}$ denote the Lipschitz constant associated with the MLP of the k-th neighborhood aggregation layer of GIN-0. We assume that all the nodes of the graph are initially annotated with a single feature equal to 1. Then, we have:

$$
\begin{aligned}
||\mathbf{h}_v^{(1)} - \mathbf{h}_u^{(1)}||_2 &= \left|\left|\text{MLP}^{(1)}\left(\sum_{w\in\mathcal{N}^+(v)}\mathbf{h}_w^{(0)}\right) - \text{MLP}^{(1)}\left(\sum_{w\in\mathcal{N}^+(u)}\mathbf{h}_w^{(0)}\right)\right|\right|_2 \\
&\leq L_f^{(1)}\left|\left|\sum_{w\in\mathcal{N}^+(v)} 1 \quad - \sum_{w\in\mathcal{N}^+(u)} 1\right|\right|_2 \\
&= L_f^{(1)}\left|\left|d(v) - d(u)\right|\right|_2
\end{aligned}
$$

$$
\begin{aligned}
||\mathbf{h}_v^{(2)} - \mathbf{h}_u^{(2)}||_2 &= \left|\left|\text{MLP}^{(2)}\left(\sum_{w\in\mathcal{N}^+(v)}\mathbf{h}_w^{(1)}\right) - \text{MLP}^{(2)}\left(\sum_{w\in\mathcal{N}^+(u)}\mathbf{h}_w^{(1)}\right)\right|\right|_2 \\
&\leq L_f^{(2)}\left|\left|\sum_{w\in\mathcal{N}^+(v)}\text{MLP}^{(1)}\left(\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(0)}\right) \quad - \sum_{w\in\mathcal{N}^+(u)}\text{MLP}^{(1)}\left(\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(0)}\right)\right|\right|_2 \\
&= L_f^{(2)}\left|\left|\text{MLP}^{(1)}\left(\sum_{w\in\mathcal{N}^+(v)}\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(0)}\right) \quad - \text{MLP}^{(1)}\left(\sum_{w\in\mathcal{N}^+(u)}\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(0)}\right)\right|\right|_2 \text{(due to Lemmas 3, 4)} \\
&\leq L_f^{(2)}L_f^{(1)}\left|\left|\sum_{w\in\mathcal{N}^+(v)}\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(0)} \quad - \sum_{w\in\mathcal{N}^+(u)}\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(0)}\right|\right|_2 \\
&= L_f^{(2)}L_f^{(1)}\left|\left|\sum_{w\in\mathcal{N}^+(v)}\sum_{x\in\mathcal{N}^+(w)} 1 \quad - \sum_{w\in\mathcal{N}^+(u)}\sum_{x\in\mathcal{N}^+(w)} 1\right|\right|_2 \\
&\vdots \\
||\mathbf{h}_v^{(K)} - \mathbf{h}_u^{(K)}||_2 &= \left|\left|\text{MLP}^{(K)}\left(\sum_{w\in\mathcal{N}^+(v)}\mathbf{h}_w^{(K-1)}\right) - \text{MLP}^{(K)}\left(\sum_{w\in\mathcal{N}^+(u)}\mathbf{h}_w^{(K-1)}\right)\right|\right|_2 \\
&\leq L_f^{(K)}\left|\left|\sum_{w\in\mathcal{N}^+(v)}\text{MLP}^{(K-1)}\left(\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(K-2)}\right) \quad - \sum_{w\in\mathcal{N}^+(u)}\text{MLP}^{(K-1)}\left(\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(K-2)}\right)\right|\right|_2 \\
&= L_f^{(K)}\left|\left|\text{MLP}^{(K-1)}\left(\sum_{w\in\mathcal{N}^+(v)}\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(K-2)}\right) - \text{MLP}^{(K-1)}\left(\sum_{w\in\mathcal{N}^+(u)}\sum_{x\in\mathcal{N}^+(w)}\mathbf{h}_x^{(K-2)}\right)\right|\right|_2 \text{(due to Lemmas 3, 4)} \\
&\leq L_f^{(K)}L_f^{(K-1)}\dots L_f^{(2)}\left|\left|\sum_{w\in\mathcal{N}^+(v)}\sum_{x\in\mathcal{N}^+(w)}\dots \quad \text{MLP}^{(1)}\left(\sum_{z\in\mathcal{N}^+(y)}\mathbf{h}_z^{(0)}\right)\right.\right. \\
&\qquad\qquad \left.\left. - \sum_{w\in\mathcal{N}(u)\cup\{u\}}\sum_{x\in\mathcal{N}^+(w)}\dots \quad \text{MLP}^{(1)}\left(\sum_{z\in\mathcal{N}^+(y)}\mathbf{h}_z^{(0)}\right)\right|\right|_2 \\
&= L_f^{(K)}L_f^{(K-1)}\dots L_f^{(2)}\left|\left|\text{MLP}^{(1)}\left(\sum_{w\in\mathcal{N}^+(v)}\sum_{x\in\mathcal{N}^+(w)}\dots\sum_{z\in\mathcal{N}^+(y)}\mathbf{h}_z^{(0)}\right)\right.\right. \\
&\qquad\qquad \left.\left. - \text{MLP}^{(1)}\left(\sum_{w\in\mathcal{N}(u)\cup\{u\}}\sum_{x\in\mathcal{N}^+(w)}\dots\sum_{z\in\mathcal{N}^+(y)}\mathbf{h}_z^{(0)}\right)\right|\right|_2 \text{(due to Lemmas 3, 4)}
\end{aligned}
$$

$$\leq L_f^{(K)} L_f^{(K-1)} \dots L_f^{(1)} \Bigg\| \sum_{w \in \mathcal{N}^+(v)} \sum_{x \in \mathcal{N}^+(w)} \dots \sum_{z \in \mathcal{N}^+(y)} \mathbf{h}_z^{(0)} - \sum_{w \in \mathcal{N}^+(u)} \sum_{x \in \mathcal{N}^+(w)} \dots \sum_{z \in \mathcal{N}^+(y)} \mathbf{h}_z^{(0)} \Bigg\|_2$$

$$= L_f^{(K)} L_f^{(K-1)} \dots L_f^{(1)} \Bigg\| \sum_{w \in \mathcal{N}^+(v)} \sum_{x \in \mathcal{N}^+(w)} \dots \sum_{z \in \mathcal{N}^+(y)} 1 - \sum_{w \in \mathcal{N}^+(u)} \sum_{x \in \mathcal{N}^+(w)} \dots \sum_{z \in \mathcal{N}^+(y)} 1 \Bigg\|_2$$

We observe that for $k = 1$, $d(v)$ and $d(u)$ are equal to the number of walks of length 1 starting from nodes v and u, respectively. Likewise, for $k = 2$, $\sum_{w \in \mathcal{N}(v) \cup \{v\}} \sum_{x \in \mathcal{N}(w) \cup \{w\}} 1$ and $\sum_{w \in \mathcal{N}(u) \cup \{u\}} \sum_{x \in \mathcal{N}(w) \cup \{w\}} 1$ are equal to the number of walks of length 2 starting from nodes v and u, respectively. And more generally, for $k = K$, $\sum_{w \in \mathcal{N}(v) \cup \{v\}} \sum_{x \in \mathcal{N}(w) \cup \{w\}} \dots \sum_{z \in \mathcal{N}(y) \cup \{y\}} 1$ and $\sum_{w \in \mathcal{N}(u) \cup \{u\}} \sum_{x \in \mathcal{N}(w) \cup \{w\}} \dots \sum_{z \in \mathcal{N}(y) \cup \{y\}} 1$ are equal to the number of walks of length K starting from nodes v and u, respectively. Thus, the representations learned by GIN-0 are related to the number of walks emanating from each node.

B Experimental Setup

In all the experiments performed on the ENZYMES, IMDB-BINARY and IMDB-MULTI, Cora and Citeseer datasets, the different GNN models were trained in the original learning tasks (graph classification for ENZYMES, IMDB-BINARY and IMDB-MULTI and node classification for Cora and Citeseer). For datasets where nodes are annotated with some initial features (e. g., ENZYMES, Cora, Citeseer), those features were not taken into account. Each dataset is randomly split into training, validation, and test sets with an $80 : 10 : 10$ split ratio, respectively. All models consist of a series of neighborhood aggregation layers. For node-level tasks, the final neighborhood aggregation layer is followed by a 2-layers MLP which produces class probabilities. For graph-level tasks, the final neighborhood aggregation layer is followed by a readout function which computes the sum of the representations of the nodes. The output of the readout function is passed on to a 2-layers MLP which produces class probabilities. For all experiments, the batch size is set equal to 64. Each model is trained for 300 epochs by minimizing the cross-entropy loss. We use the Adam optimizer for model training. We store in the disk the parameters of the model that achieved the lowest validation loss and those parameters are retrieved once training has finished. Then, 100 nodes from the graphs that belong to the test set are randomly sampled and the representations of those nodes are extracted from the final neighborhood aggregation layer. Then, pairwise distances of those nodes are computed and compared against the corresponding distances that emerge from the (normalized) walks.

C Additional Visualizations

We provide further experimental results in Figures 9, 10 and 11. The three Figures compare the Euclidean distance of the representations of the nodes that emerge at the third layer of GIN-0 (resp. GCN) against the Euclidean distance of the number of walks (resp. sum of normalized walks) of length 3 starting from those nodes on the IMDB-MULTI, Cora and Citeseer datasets, respectively.

Figure 12 illustrates how the Euclidean distance between the representations of two nodes at the third layer of GIN-0 and GCN varies as a function of the descrease in the number of walks due to the removal of edges from the node's local neighborhood. Each perturbed graph emerges from the original graph by just removing one edge which connects two nodes whose shortest path distance from v is at most 2. We denote by u the node of the perturbed graph that corresponds to node v of the original graph.

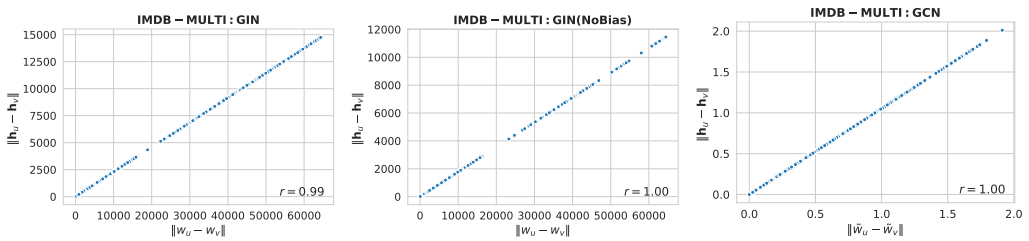

Figure 9: Euclidean distances of the representations generated at the third layer of the different models vs. Euclidean distances of the number of walks (or normalized walks) of length 3 starting from the different nodes on the IMDB-MULTI dataset. Nodes are annotated with a single feature equal to 1.

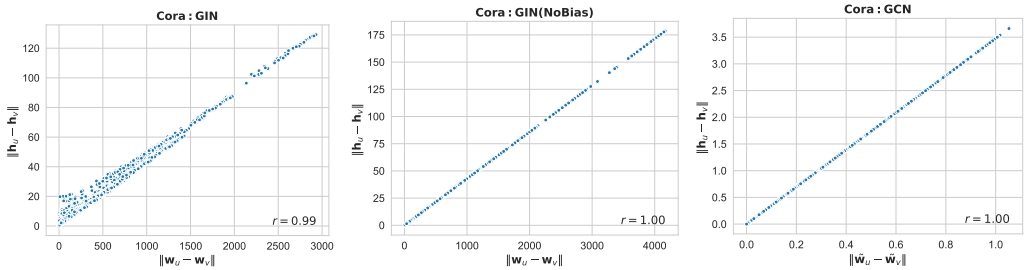

Figure 10: Euclidean distances of the representations generated at the third layer of the different models vs. Euclidean distances of the number of walks (or normalized walks) of length 3 starting from the different nodes on the Cora dataset. Nodes are annotated with a single feature equal to 1.

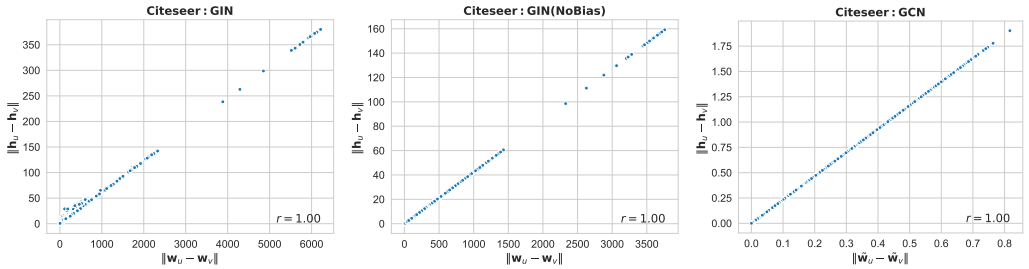

Figure 11: Euclidean distances of the representations generated at the third layer of the different models vs. Euclidean distances of the number of walks (or normalized walks) of length 3 starting from the different nodes on the Citeseer dataset. Nodes are annotated with a single feature equal to 1.

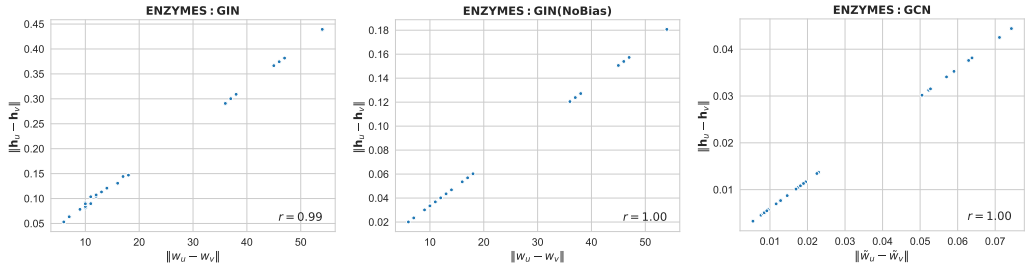

Figure 12: Euclidean distances of the representations generated at the third layer of the different models vs. Euclidean distances of the number of walks (or sum of normalized walks) of length 3 starting from the different nodes. Nodes v and u correspond to the same node in the original and the perturbed graph, respectively. Each perturbed graph has emerged by removing one edge from node v's neighborhood.