# United We Stand, Divided We Fall: Networks to Graph (N2G) Abstraction for Robust Graph Classification under Graph Label Corruption

**Zhiwei Zhen**
University of Texas at Dallas
zhiwei.zhen@utdallas.edu

**Yuzhou Chen**
Temple University
yuzhou.chen@temple.edu

**Murat Kantarcioglu**
University of Texas at Dallas
muratk@utdallas.edu

**Kangkook Jee**
University of Texas at Dallas
kangkook.jee@utdallas.edu

**Yulia R. Gel**
University of Texas at Dallas
National Science Foundation
ygl@utdallas.edu

## Abstract

Nowadays, graph neural networks (GNN) are the primary machinery to tackle (semi)-supervised graph classification tasks. The aim here is to predict classes for unlabeled graphs, given a collection of graphs with known labels. However, in many real-world applications, the available information on graph classes may be distorted either due to incorrect labeling process (e.g., as in biochemistry and bioinformatics) or may be subject to targeted attacks (e.g., as in network-based customer attrition analytics). Over the past few years, the increasing number of studies has indicated that GNNs are prone both to noisy node and noisy graph labels, and while this problem has received noticeable attention for node classification tasks, vulnerability of GNNs for graph classification with perturbed graph labels still remains in its nascence. We hypothesize that this challenge can be addressed by the universal principle *United We Stand, Divided We Fall*. In particular, most GNNs view each graph as a standalone entity and, as a result, are limited in their abilities to account for complex interdependencies among the graphs. Inspired by the recent studies on molecular graph learning, we propose a new robust knowledge representation called *Networks to Graph* (N2G). The key N2G idea is to construct a new abstraction where each graph in the collection is now represented by a node, while an edge then reflects some sort of similarity among the graphs. As a result, the graph classification task can be then naturally reformulated as a node classification problem. We show that the proposed N2G representation approach does not only improve classification performance both in binary and multi-class scenarios but also substantially enhances robustness against noisy labels in the training data, leading to relative robustness gains up to 11.7% on social network benchmarks and up to 25.8% on bioinformatics graph benchmarks under 10% of graph label corruption rate.

## 1 Introduction

Graph classification tasks spread a wide range of application domains, from analysis of social networks to molecular structures, to cyber-security systems, and to financial risk management [28, 37, 44]. Currently, Graph Neural Networks (GNNs) are adopted as the primary state-of-the-art methodology for graph classification. By incorporating the information on the structure space into the learning process, GNNs allow for achieving significant gains in graph classification performance in comparison with, for example, kernel-based techniques and other more traditional machine learning tools. However, despite its competitive performance, one of the main limitations of GNNs is sensitivity to various types of label perturbations. This issue has been recently observed in the context of node classification in graphs [4, 5, 26, 42], and various techniques, including co-teaching and loss correction have been developed to address noisy node labels in a given graph. However, the sensitivity of GNN-based models to label perturbations in the case of graph classification remains in its nascence [23]. We show that GNN-based models for graph classification are also prone to label noise in the training data. In particular, having wrong labels for the 10% of the randomly chosen training data, in some cases, may result in up to 20% reduction in overall test accuracy (that is, the test accuracy reduces from 80.73% for clean graphs to 60.24% for perturbed graphs). Recently, Dai et al. [4] argue that the conventional tools developed for addressing noisy labels for the node classification tasks, such as loss correction techniques [25], co-teaching [40] and various forms of sample selection, may not be directly applicable to the classification of graphs with perturbed labels. Indeed, approaches, such as loss correction, assume that we classify independent and identically distributed entities, for example, spatial processes in $\mathbb{R}^2$ for image dataset. However, in the case of classification of such complex objects as graphs, it requires knowledge of the distribution of the graph as a random object which involves often unrealistic assumptions on (joint) degree distribution and distribution of higher-order graph structural properties (e.g., motifs) within the restrictive stochastic block models and random dot product graph models. Clearly, even less can be said on systematic validation of such distributional assumptions. In turn, the assumption of independence among graphs is even more problematic. In many domains such as cybersecurity, system provenance graphs generated by observing program execution may contain common subgraphs (e.g, MS Windows boot sequence) violating independence assumption.

Inspired by the recent results on molecular graph learning [13, 30, 41], we propose a new robust knowledge representation called *Networks to Graph* (N2G). Our key idea is motivated by the motto *United We Stand, Divided We Fall*. That is, we propose to view a sequence of graphs to be classified not as standalone entities with some structural and feature similarities as the current methods do, but rather as a single graph-structured object where nodes correspond to the original graphs and edges among them reflect some sort of similarity. As a result, our proposed N2G abstraction now re-formulates the graph classification problem into a node classification task. We find that the N2G not only leads to comparable or competitive performance on clean graphs but, most importantly, results in substantial gains in robustness against various sorts of graph label noise in the training data. Such phenomena in the robustness gains can be explained by the fact that N2G allows us to explicitly integrate various types of interdependencies among the original graphs, including higher-order polyadic relations, into graph classification tasks performed by GNNs. In contrast, the conventional approaches are based on utilizing GNNs to learn the key similarity in each graph individually, without accounting for graph interrelations. As such the conventional methods are more prone to label perturbations (i.e., *Divided We Fall)*, while N2G, viewing the set of graphs as a *single unified* object which explicitly describes complex interdependencies, is substantially more resistant to label perturbations (i.e., *United We Stand*).

**Why N2G Works?** The intuition behind N2G can be also explained through $k$-nearest neighbor approaches (kNN). Suppose that each graph is associated with a person. We can train a model to classify people based on their individual socio-demographic characteristics. Alternatively, we can check how these people are related to each other, create a social network, and now classify people, while accounting for their implicit and explicit similarities or relationships. Classification of each individual is then no longer based only on his/her socio-demographic characteristics but also on those of his/her peer circle (i.e., nearest neighbors). As shown by [33], robustness properties of kNN inherently depend on the value of $k$ (i.e., the depth of the peer circle). That is, the kNN classifier tends to be non-robust for lower values of $k$ and its robustness increases with fast-growing $k$, approaching that of the Bayes Optimal classifier. In this sense, N2G may be viewed as inheriting kNN robustness properties for higher values of $k$ (i.e., deeper peer circle, for example, associated with extended family, all friends and colleagues), since any graph (person) to be classified employs information from **all** its neighbors with $k$ being the maximum possible. Finally, the N2G approach is also inherently connected to *statistical relational learning* (SRL) which aims to model a joint distribution over relational data and often is represented via probabilistic graphical models such as Bayesian networks [9, 14, 15]. However, in contrast to SRL, the focus of N2G is not on learning probability models or logic reasoning.

The significance of our contributions can be summarized as follows:

- We investigate a largely unexplored problem of the impact of noisy graph labels on graph classification performance delivered by graph neural networks.
- We propose a new robust knowledge representation for graph learning under noisy label scenarios which explicitly accounts for complex interrelations among multiple graphs in a systematic and unified manner.
- We show the proposed N2G abstraction results not only in on par or more competitive performance on clean graphs (both in binary and multi-class cases) but, most importantly, leads to substantially higher resistance to graph label noise in training data, with relative gains up to 11.7% on social network benchmarks and up to 25.8% on bioinformatics graph benchmarks under 10% of graph label corruption rate.

## 2 Related Work

**Graph of Graphs Representations** The idea of integrating standalone entities into a single joint complex entity is quite universal and spans a broad range of domains, from the System of Systems (SoS) concept in engineering to the graph of graphs (GoG) in biochemistry and biomedicine. In particular, in the last few years, this approach has been combined with GNNs to predict chemical compounds [13]. The idea here is to form a hierarchical structure of compound graphs and an inter-compound graph, integrated into a single network, which allows for learning both types of graphs in an end-to-end manner. Also, in a hierarchical manner, GoGNN of [30] focuses on learning local and global entity structure with the aim to predict chemical-chemical interactions and drug-drug interactions. This idea is further extended by [31] for graph classification tasks, based on exploiting two GNNs, one for for learning local entities and another one for learning structural interactions. Edges in the interaction graph are formed if two local graphs share some joint information, for example, two proteins are linked in the interaction graph if they are either connected or share at least two common neighbors. In turn, [18] propose a semi-supervised graph learning approach SEAL and the associated embedding procedure SAGE that first, embeds graph instances into vectors and then adaptively update classifiers based on the selected most informative instances. Finally, [41] construct a heterogeneous motif graph based on the motif-level relationships in smaller molecular graphs, where each motif node is a motif in the vocabulary, each molecular node is a molecule, and there are two types of links: motif-molecule and motif-motif edges. That is, loosely, speaking smaller molecular graphs are connected if they share motif structure. The idea is further exploited by [34] for graph classification under imbalanced settings, where the resulting model $\mathrm{G}^2\mathrm{GNN}$ aggregates information from the neighboring graphs which are defined as the top $k$ topologically similar graphs via kNN. All of the considered approaches assume that the labels are trustworthy.

Another relevant approach called SPADE [2] presents a black-box method for assessing adversarial robustness for robust image classification. The idea of SPADE is based on the following steps: 1) reshaping each image into a vector representation as an input sample and extracting a vector representation before the softmax layer per image as the output sample; 2) forming input and output graphs based on the input and output samples, respectively, using kNN; 3) examining the bijective distance mappings between the input and output graph-based manifolds. In particular, the largest generalized eigenvalue computed with the Laplacians of the input and output graphs, called the SPADE score can be used to quantify the adversarial robustness, with higher scores implying higher levels of vulnerability. The approach of SPADE can be extended to graphs and also combined with N2G, including simultaneous integration of multiple graph distances via supra-Laplacian.

To the best of our knowledge, our N2G is the first time the GoG idea of integrating multiple standalone entities into a joint network has been introduced to robustify graph learning under label perturbations. Furthermore, as our numerical experiments show the non-hierarchical model architecture, with a wide range of attributes from smaller graphs, rather than only from particular motifs or a subset of joint nodes/edges, tends to deliver the most robust and stable performance under label perturbations. This phenomenon is intuitive: the basic constructions are often more stable than more sophisticated ones.

Finally, there exists a stream of studies in network sciences on the so called *network of networks* [3]; however, the nature of these concepts differs from the ideas discussed above in the sense that the focus there is on the initially highly interconnected systems in the form of multilayer or multiplex graphs, e.g., cyber-physical infrastructures such as highly interdependent power, transportation, and communications networks, rather than on integrating multiple standalone entities into a joint system based on some sort of similarity.

**Graph Neural Networks for Graph Classification under Label Perturbations** GNNs is currently the primary machinery for both node and graph classification [28, 37, 38]. However, as the increasing number of recent studies demonstrate [7, 19], GNNs tend to be vulnerable to corrupted node and graph labels, which often results in substantial performance losses and overfitting [43]. Most existing tools addressing GNNs for noisy labels focus on the node classification tasks. Such techniques include, e.g., loss correction [10, 21], pseudo label miner [4], and sample selection [40]. However, such remedial techniques for corrupted node labels tend to rely on the restrictive distributional assumptions on the observed data which are particularly challenging to verify for graphs, thereby making it harder to extend these approaches to graph classification tasks. In general, the problem of GNN sensitivity to corrupted graph labels and the associated remedial techniques remains largely unexplored. To the best of our knowledge, the only currently available approaches are loss correction for graph classification tasks [23] and G-Mixup by [12]. In the case of G-Mixup, the idea is to view two graph training sets as following two different graphon models and then to mix up these two graphons into a joint mixed graphon which can then be used to generate synthetic graphs, aiming to improve generalization and robustness of GNNs. This paper aims to further fill this gap and offers a new robust N2G representation learning scheme with small-loss tricks, addressing corrupted labels in the training set for graph classification tasks.

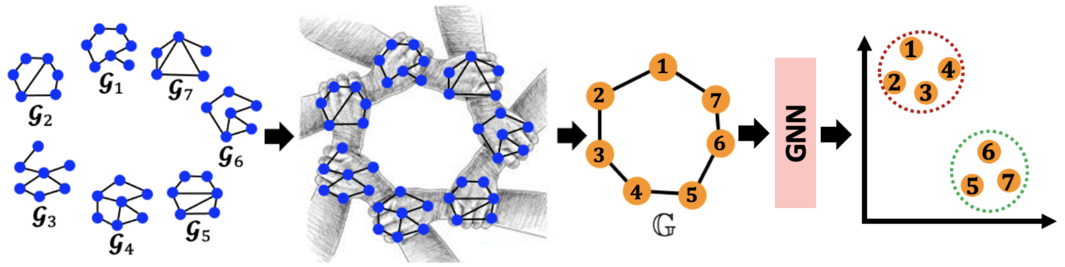


**Figure 1:** An illustration of N2G representation learning framework.

## 3 The N2G Knowledge Representation

Let $\tilde{\mathcal{G}} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$ be a set of graphs, where $N$ is the number of observed graphs. For each graph $\mathcal{G}_i = \{\mathcal{V}_i, \mathcal{E}_i\}$, let $\mathcal{V}_i = \{v^i_1, v^i_2, \ldots, v^i_{p_i}\}$ be a set of nodes and $\mathcal{E}_i$ be the set of edges. Some of the observed graphs in $\tilde{\mathcal{G}}$ are also associated with the attached categorical label, while the labels of other graphs are unknown. Here we primarily focus on graph classification with perturbed training labels. That is, suppose that we are given a training set $\tilde{\mathcal{G}}_{\text{train}}$ consisting of $N_{\text{train}}$ graphs in $\tilde{\mathcal{G}}$ with perturbed labels $l^{(s)}_*$, $s = 1, 2, \ldots, N_{\text{train}}$. Our goal is to build a model that learns a function $\mathcal{H}(\{\mathcal{G}^{(s)}\}_{s=1}^{N_{\text{train}}})$ which predicts unknown labels for graphs in $\tilde{\mathcal{G}} \setminus \tilde{\mathcal{G}}_{\text{train}}$ (please refer to Appendix I for the notation table). Algorithm 1 (in Appendix A) shows more details of our N2G algorithm.

### 3.1 The N2G Methodology

To address the sensitivity of GNNs to noisy graph labels, we construct a new graph-structured object $\mathbb{G}$ called Network of Graphs (N2G). In particular, $\mathbb{G}$ is such that its node set $\mathbb{V}$ is formed by $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$. In turn, an edge $e_{ij}$ between $\mathcal{G}_i$ and $\mathcal{G}_j$, $i, j = 1, \ldots, N$, is defined by how similar $\mathcal{G}_i$ and $\mathcal{G}_j$ are in terms of some suitable graph similarity metric. That is, let $\mathscr{D}_{ij} = \mathscr{D}(\mathcal{G}_i, \mathcal{G}_j)$ be a distance between graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ and let $\tau > 0$ be a user-selected threshold (In our experiments we select $\tau$ from the quantiles of the empirical distribution of $\mathscr{D}_{ij}$, $i, j = 1, \ldots, N$, using the cross-validation argument. We present a sensitivity study to the choice of $\tau$ in Appendix D.). Then we get the following equation:

$$e_{ij} = \begin{cases} 0, & \text{if } \mathscr{D}(\mathcal{G}_i, \mathcal{G}_j) > \tau, \\ \mathscr{D}(\mathcal{G}_i, \mathcal{G}_j), & \text{if } \mathscr{D}(\mathcal{G}_i, \mathcal{G}_j) \leq \tau, \end{cases} \tag{1}$$

where smaller values of $\mathscr{D}(\mathcal{G}_i, \mathcal{G}_j)$ imply higher similarity among $\mathcal{G}_i$ and $\mathcal{G}_j$). That is, each $\mathcal{G}_i$ is now a (super)-node in the N2G object $\mathbb{G}$, and the set of weighted edges $e_{ij}$ is the edge set $\mathbb{E}$ of $\mathbb{G}$ (see Appendix A for the list of the considered graph similarity measures). Fig. 1 presents the flowchart of the N2G idea. Furthermore, for each (super)-node $i \in \mathbb{G}$ (that is, for each graph $\mathcal{G}_i$), we also obtain its node features $\mathbb{X}_i$ (an $N \times F$ matrix) through calculating the network statistics of the graph $\mathcal{G}_i$, where $F$ is the dimension of the node features. Here we use 7 network statistics: average degree centrality, betweenness centrality, closeness centrality, eigenvector centrality, current flow betweenness centrality, subgraph centrality, and current flow closeness centrality, and hence $F = 7$. For the graph classification tasks, we then feed our N2G $\mathbb{G} = \{\mathbb{V}, \mathbb{E}, \mathbb{X}\}$ into any type of GNN-based model (check Appendix B for the experimental details and Appendix F on ablation study for super-node feature $\mathbb{X}_i$).

**Nota Bene:** Our goal here is to investigate the utility of the N2G abstraction as a general tool that can be combined with **any** GNNs. As such, we focus on three (arguably) most widely used GNN architectures (GCN, GAT, GRAPHSAGE) as well as consider integration of N2G with the emerging GNNs (see Table 7 and Appendix C).

## 4 Experiments

**Datasets, Experiment Setup, and Baselines** We conduct experiments on (i) three social datasets, i.e., IMDB-BINARY, IMDB-MULTI and REDDIT-BINARY, and (ii) three biological datasets, i.e., MUTAG, BZR, and COX2. All six datasets are homogeneous datasets. The statistics of six datasets are summarized in Table 9 in Appendix B. Following the setup of [39], we use 90/10% random training/test split and provide the average accuracy in 10 runs for model evaluation. Furthermore, to evaluate the robustness of our proposed N2G framework, we perform robustness testing for all six datasets with four settings, i.e., "uniform 10%", "uniform 20%", "biased 10%" and "biased 20%" corrupted training labels. We compare our method to the four fundamental GNN architectures, namely (i) Graph Convolutional Networks [16], (ii) Graph Attention Networks (GAT) [29], (iii) GraphSAGE [11], (iv) Graph Isomorphism Network (GIN) [39], as well as (v) the recent graph data augmentation approach G-Mixup by [12]. Furthermore, we also integrate N2G algorithm into other state-of-the-art GNN-based models (see Appendix B for more details of additional state-of-the-art GNN-based models). To make a fair comparison, we re-produce the above GNNs on all datasets and follow their settings by using the same optimized hyperparameters and experimented datasets (including train/test splits). All experiments are conducted on one NVIDIA GetForce RTX 3090. For dataset and coding details, please check `https://anonymous.4open.science/r/N2G-6443/README.md`.

**The N2G Setup** We implement our N2G based on existing vanilla GNN architecture, i.e., applying a vanilla GNN-based model on N2G instead of all graphs in the dataset. We consider three types of GNNs based on three graph comparison metrics (see Section 3), including (i) Vertex-edge distance [1], (ii) Lambda distance [6], and (iii) DeltaCon distance with $\epsilon = 0.3$. Hence, for each GNN, we have ***[GNN Name]-[N2G]-[Graph Comparison Metric]***. For instance, N2G-GCN-VERTEX refers to applying the GCN model on N2G, which is built upon the vertex-edge overlap distance metric. Note that we ensure both the original and modified architectures have approximately the same number of parameters. More details about datasets, experiments setup, and hyperparameters are in Appendix B.

**Corrupted Labels** Similar to [23], we assume that there is a noise process $N$ corrupting the training labels, where $N_{a,b}$ is the probability label $a$ corrupted to label $b$. We further assume that there are 2 kinds of corrupted labels : (i) Uniformly corrupted and (ii) Biased corrupted labels. For uniformly corrupted labels, the label noise is uniformly randomly distributed across the whole training dataset i.e., $N_{a,b} = N_{c,d}$ for all $a, b, c, d \in \{1, 2, \ldots, C\}$, where $C$ is the number of labels classes. For biased corrupted labels, the noise corrupts one label class i.e., for a certain class $a$, $N_{a,b} = N_{a,c} \neq 0$ for all $b, c \in \{1, 2, \ldots, C\}$, $N_{a^*,b} = 0$ for all $a^* \neq a$, $b \in \{1, 2, \ldots, C\}$, which aims to reflect a real world scenario when one of the subpopulations is discriminated due to various types of subconscious and other social biases. Without loss of generality, we apply the biased label noise to class "0".

### 4.1 N2G Results on Bioinformatics Graphs

Table 1 reports graph classification results in terms of the mean accuracy and standard deviation on clean graphs and graphs with noisy labels on three biochemical networks (i.e., "Uniform$_{10}$" and "Uniform$_{20}$"). The results of our N2G-based model and baselines with biased noisy labels are summarized in Table 2. Table 1 indicates that our proposed N2G outperforms all competing models on all three datasets, both for clean and corrupted graphs. More specifically, for clean graphs, N2G-GCN-VERTEX can improve upon GCN by a margin of 10.33%, 8.36%, and 10.36% on MUTAG, BZR, and COX2, respectively. In turn, for corrupted graphs with uniform 10% noisy

**Table 1:** Average accuracy on clean bioinformatics graphs and their counterparts under uniform corruption of graph labels. Corruption rates are 10% and 20%. Standard deviation is in ( ). The best results are in bold.

| Model | MUTAG | | | BZR | | | COX2 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Clean | Uniform$_{10}$ | Uniform$_{20}$ | Clean | Uniform$_{10}$ | Uniform$_{20}$ | Clean | Uniform$_{10}$ | Uniform$_{20}$ |
| GCN | 85.60 (5.80) | 70.88 (2.99) | 68.60 (4.02) | 80.49 (3.22) | 67.36 (3.74) | 60.91 (3.25) | 78.60 (1.52) | 69.09 (5.28) | 65.72 (6.25) |
| N2G-GCN-VERTEX | **94.44** (2.98) | **85.89** (3.82) | **83.40** (4.58) | **87.22** (3.71) | **84.36** (4.12) | **83.29** (3.10) | **86.74** (4.63) | **82.44** (4.07) | **80.17**(3.09) |
| N2G-GCN-LAMBDA | 92.58 (4.85) | 82.81 (3.55) | 81.41 (3.08) | 85.19 (1.93) | 81.76 (3.30) | 80.94 (2.66) | 84.07 (1.69) | 80.74 (3.99) | 78.01 (4.94) |
| N2G-GCN-DELTACON | 88.89 (3.62) | 81.90 (4.22) | 79.81 (3.94) | 86.50 (3.96) | 81.55 (4.67) | 80.44 (2.37) | 84.26 (3.66) | 79.85 (4.91) | 78.63 (4.17) |
| GAT | 87.40 (5.31) | 65.42 (3.11) | 58.33 (4.31) | 83.21 (4.52) | 70.24 (4.97) | 62.71 (3.22) | 79.26 (3.54) | 65.32 (7.76) | 61.67 (2.49) |
| N2G-GAT-VERTEX | **94.07** (1.88) | **84.39** (3.92) | **83.47** (3.58) | **88.09** (2.43) | **86.72** (2.91) | **83.28** (2.91) | **88.26** (3.17) | **83.72** (2.91) | **81.95** (3.42) |
| N2G-GAT-LAMBDA | 90.71 (2.01) | 84.07 (5.23) | 82.19 (4.10) | 84.21 (1.79) | 82.90 (4.25) | 81.02 (3.55) | 85.34 (3.23) | 82.07 (5.23) | 80.26 (1.78) |
| N2G-GAT-DELTACON | 92.50 (3.63) | 82.83 (5.30) | 79.55 (4.33) | 85.97 (3.29) | 83.47 (4.48) | 82.09 (1.97) | 84.69 (3.19) | 82.08 (5.87) | 81.04 (4.04) |
| GRAPHSAGE | 85.73 (4.70) | 77.11 (3.52) | 70.70 (3.71) | 77.53 (3.73) | 70.15 (6.06) | 65.48 (2.39) | 79.01 (2.42) | 63.21 (6.73) | 59.74 (3.37) |
| N2G-GRAPHSAGE-VERTEX | 91.97 (1.92) | 82.10 (4.28) | 79.53 (3.35) | 89.14 (2.47) | 82.68 (4.94) | 78.03 (3.95) | 83.93 (3.60) | 81.07 (5.54) | 80.47 (2.55) |
| N2G-GRAPHSAGE-LAMBDA | **92.49** (2.19) | **82.30** (2.69) | 78.58 (5.67) | **89.40** (1.83) | **85.19** (2.41) | **82.94** (1.87) | **86.88** (4.51) | **83.93** (3.60) | **81.70** (3.17) |
| N2G-GRAPHSAGE-DELTACON | 88.89 (2.93) | 82.19 (5.58) | **80.15** (3.67) | 87.61 (2.55) | 84.42 (4.85) | 80.19 (5.75) | 84.03 (2.33) | 81.36 (4.06) | 79.05 (3.65) |
| GIN | 89.01 (6.02) | 77.37 (5.54) | 71.05 (7.89) | 85.63 (2.59) | 73.49 (3.79) | 68.93 (4.15) | 83.51 (4.75) | 73.49 (3.79) | 68.93 (4.15) |
| G-Mixup | 87.54 (2.45) | 77.25 (4.11) | 71.22 (2.99) | 80.11 (2.94) | 74.82 (1.34) | 69.54 (2.54) | 80.23 (3.29) | 71.74 (3.25) | 63.57 (2.91) |

**Table 2:** Average accuracy (%) on bioinformatics graphs under biased corruption of graph labels. Standard deviation is in ( ). The best results are in bold.

| Model | MUTAG | | BZR | | COX2 | |
|---|---|---|---|---|---|---|
| | Biased$_{10}$ | Biased$_{20}$ | Biased$_{10}$ | Biased$_{20}$ | Biased$_{10}$ | Biased$_{20}$ |
| GCN | 68.95 (4.36) | 68.16 (2.82) | 64.61 (3.45) | 57.16 (3.27) | 66.91 (4.82) | 60.52 (5.58) |
| N2G-GCN-VERTEX | **85.89** (3.82) | **82.44** (4.22) | **81.42** (2.92) | **80.38** (3.09) | **79.84** (3.71) | **76.78** (4.03) |
| N2G-GCN-LAMBDA | 80.56 (4.83) | 80.19 (4.22) | 81.07 (2.83) | 77.48 (3.08) | 77.84 (4.54) | 75.32 (3.59) |
| N2G-GCN-DELTACON | 81.20 (2.82) | 78.74 (4.95) | 80.04 (3.97) | 76.89 (3.71) | 77.20 (3.19) | 75.68 (4.22) |
| GAT | 66.72 (4.14) | 64.52 (2.59) | 65.96 (3.47) | 59.85 (1.85) | 64.52 (4.93) | 62.73 (5.82) |
| N2G-GAT-VERTEX | **84.46** (3.52) | **82.19** (2.36) | **82.43** (3.55) | **80.15** (4.10) | **82.47** (1.29) | **78.90** (4.06) |
| N2G-GAT-LAMBDA | 82.91 (3.50) | 80.43 (3.67) | 81.19 (4.38) | 79.55 (3.75) | 80.05 (4.65) | 77.06 (3.73) |
| N2G-GAT-DELTACON | 84.43 (4.63) | 81.54 (5.74) | 81.94 (5.49) | 78.63 (4.20) | 80.73 (1.98) | 77.85 (2.77) |
| GRAPHSAGE | 68.42 (9.45) | 61.58 (7.14) | 67.30 (2.47) | 65.61 (5.79) | 68.54 (5.29) | 64.01 (2.17) |
| N2G-GRAPHSAGE-VERTEX | 82.43 (4.45) | 80.48 (4.65) | 81.97 (3.56) | 78.88 (3.52) | 79.03 (4.91) | 77.84 (3.43) |
| N2G-GRAPHSAGE-LAMBDA | **84.39** (3.85) | **82.71** (4.84) | **82.43** (5.25) | **80.26** (4.71) | **81.53** (2.82) | **77.43** (4.63) |
| N2G-GRAPHSAGE-DELTACON | 84.32 (4.43) | 79.40 (5.47) | 81.44 (5.49) | 78.95 (3.53) | 79.28 (3.80) | 76.03 (4.06) |
| GIN | 69.18 (4.33) | 61.05 (4.74) | 70.15 (4.10) | 61.85 (5.07) | 70.11 (1.98) | 65.19 (4.44) |
| G-Mixup | 72.44 (1.98) | 68.79 (2.87) | 73.45 (1.38) | 68.34 (2.54) | 68.97 (3.44) | 59.03 (6.53) |

**Table 3:** Average accuracy (%) on clean social networks and their counterparts with uniformly corrupted graph labels. Corruption rates are 10% and 20%. Standard deviation is in ( ). The best results are in bold.

| Model | IMDB-BINARY | | | IMDB-MULTI | | | REDDIT-BINARY | | |
|---|---|---|---|---|---|---|---|---|---|
| | Clean | Uniform$_{10}$ | Uniform$_{20}$ | Clean | Uniform$_{10}$ | Uniform$_{20}$ | Clean | Uniform$_{10}$ | Uniform$_{20}$ |
| GCN | 78.18 (2.16) | 64.11 (6.89) | 60.32 (8.67) | 49.33 (3.21) | 43.23 (1.08) | 40.50 (5.69) | 82.37 (6.28) | 75.33 (4.78) | 69.54 (4.39) |
| N2G-GCN-VERTEX | **80.36** (5.97) | **74.82** (5.05) | **70.33** (5.90) | **56.81** (5.05) | **50.00** (2.59) | **47.01** (4.88) | **86.28** (2.95) | **80.13** (4.78) | **76.91** (4.71) |
| N2G-GCN-LAMBDA | 78.43 (3.34) | 71.80 (3.03) | 68.93 (5.90) | 51.81 (5.05) | 45.05 (5.44) | 42.01 (4.16) | 83.38 (4.91) | 78.35 (5.98) | 73.92 (4.67) |
| N2G-GCN-DELTACON | 79.61 (5.91) | 70.50 (4.48) | 67.60 (7.67) | 52.74 (3.47) | 48.78 (4.94) | 45.45 (5.09) | 83.02 (4.97) | 77.13 (6.78) | 72.27 (3.97) |
| GAT | 70.51 (4.92) | 66.92 (8.89) | 60.30 (9.67) | 47.89 (2.55) | 44.20 (4.48) | 41.52 (5.67) | 80.37 (3.79) | 74.61 (4.36) | 65.19 (6.71) |
| N2G-GAT-VERTEX | **73.18** (5.13) | **69.83** (5.01) | **67.41** (6.77) | **49.91** (5.66) | **47.52** (4.25) | **45.67** (3.73) | **84.19** (6.37) | **80.02** (3.17) | **76.17** (2.98) |
| N2G-GAT-LAMBDA | 71.22 (2.75) | 67.91 (4.21) | 64.02 (4.19) | 49.16 (4.89) | 45.33 (4.72) | 43.51 (5.18) | 82.75 (4.91) | 77.33 (5.98) | 73.46 (4.67) |
| N2G-GAT-DELTACON | 70.92 (3.48) | 67.30 (6.15) | 63.30 (5.95) | 48.93 (3.75) | 44.67 (4.65) | 41.89 (2.41) | 81.78 (5.17) | 76.97 (3.82) | 74.09 (3.79) |
| GRAPHSAGE | 72.31 (5.35) | 66.91 (6.02) | 64.10 (5.55) | 48.40 (5.94) | 45.61 (1.76) | 43.73 (3.36) | 85.21 (3.31) | 76.52 (3.07) | 68.40 (5.50) |
| N2G-GRAPHSAGE-VERTEX | 75.33 (4.64) | 68.62 (3.74) | 65.98 (4.11) | 51.32 (3.24) | 49.65 (3.28) | 46.75 (2.75) | 85.43 (5.19) | 77.44 (4.80) | 71.81 (5.70) |
| N2G-GRAPHSAGE-LAMBDA | **78.36** (5.97) | **73.33** (4.64) | **68.30** (4.66) | **54.12** (3.52) | **50.85** (3.88) | **47.98** (5.12) | **87.46** (2.71) | **84.54** (3.20) | **78.94** (2.98) |
| N2G-GRAPHSAGE-DELTACON | 75.37 (6.30) | 69.50 (5.52) | 66.46 (4.56) | 51.66 (3.75) | 50.67 (4.65) | 46.89 (2.41) | 85.26 (6.01) | 78.24 (4.93) | 70.90 (5.80) |
| GIN | 72.80 (3.91) | 68.70 (4.15) | 65.17 (5.44) | 52.30 (3.82) | 46.15 (4.10) | 41.85 (5.07) | 81.00 (3.11) | 76.01 (2.96) | 70.04 (4.82) |
| G-Mixup | 79.18 (1.55) | 70.60 (4.38) | 64.70 (4.25) | 50.79 (2.72) | 46.12 (3.34) | 42.68 (4.98) | 82.82 (1.33) | 76.80 (0.99) | 70.28 (3.50) |

**Table 4:** Average accuracy (%) on social graphs under biased corruption of graph labels. Standard deviation is in ( ). The best results are in bold.

| Model | IMDB-BINARY | | IMDB-MULTI | | REDDIT-BINARY | |
|---|---|---|---|---|---|---|
| | Biased$_{10}$ | Biased$_{20}$ | Biased$_{10}$ | Biased$_{20}$ | Biased$_{10}$ | Biased$_{20}$ |
| GCN | 67.73 (3.79) | 63.95 (4.99) | 45.07 (2.80) | 42.69 (4.01) | 70.40 (3.97) | 62.08 (3.30) |
| N2G-GCN-VERTEX | **73.53** (3.51) | **70.90** (4.93) | **51.33** (3.46) | **48.89** (4.02) | **77.14** (3.17) | **75.90** (3.83) |
| N2G-GCN-LAMBDA | 70.75 (2.78) | 66.12 (1.09) | 49.55 (1.68) | 46.11 (1.02) | 75.47 (1.95) | 73.98 (2.00) |
| N2G-GCN-DELTACON | 69.85 (2.26) | 65.67 (1.15) | 47.44 (1.39) | 44.98 (1.47) | 76.10 (3.17) | 74.03 (1.67) |
| GAT | 62.64(6.87) | 54.73(1.07) | 44.68(2.76) | 39.13(1.27) | 68.64(4.76) | 62.30(5.33) |
| N2G-GAT-VERTEX | **67.33** (1.15) | **64.67** (1.57) | **47.89** (2.04) | **45.11** (3.36) | **73.08**(5.16) | **69.13**(4.93) |
| N2G-GAT-LAMBDA | 65.55(2.01) | 62.35(1.73) | 46.33 (3.71) | 41.56 (1.54) | 71.99 (3.77) | 65.89 (3.74) |
| N2G-GAT-DELTACON | 63.67(1.53) | 61.33(4.04) | 44.56(3.42) | 43.00(2.31) | 72.21(4.08) | 66.47(2.71) |
| GRAPHSAGE | 63.14(9.39) | 60.93(1.43) | 45.67(1.70) | 39.14(1.26) | 70.83(3.99) | 63.17(4.70) |
| N2G-GRAPHSAGE-VERTEX | 65.29 (4.30) | 62.53 (2.33) | 46.81(1.89) | 41.55(2.11) | 71.18(4.03) | 69.92(2.04) |
| N2G-GRAPHSAGE-LAMBDA | **68.76** (2.65) | **66.16** (3.83) | **49.08** (2.76) | **45.80** (3.01) | **73.57** (4.09) | **70.70** (3.55) |
| N2G-GRAPHSAGE-DELTACON | 68.22(3.09) | 62.17(4.00) | 46.03(3.90) | 42.11(3.44) | 70.98(3.69) | 67.93(2.44) |
| GIN | 66.47 (3.25) | 63.70 (3.98) | 44.85 (1.80) | 41.72 (4.25) | 71.52 (4.71) | 65.03 (3.77) |
| G-Mixup | 65.97 (3.99) | 62.25 (5.14) | 44.03 (2.55) | 41.90 (4.05) | 73.22 (1.58) | 65.28 (3.50) |

labels (i.e., "Uniform$_{10}$"), N2G-GCN-VERTEX can improve upon GCN by a margin of 21.18%, 25.24%, and 19.32% on MUTAG, BZR, and COX2 respectively; for corrupted graphs with uniform 20% noisy labels (i.e., "Uniform$_{20}$"), N2G-GCN-VERTEX can improve upon GCN by a margin of

**Table 5:** N2G vs. other SOTA GoGs on MUTAG with clean and uniformly corrupted graph labels. The best results are in bold.

| **Model** | **MUTAG** Clean | Uniform$_{10}$ | Uniform$_{20}$ |
|---|---|---|---|
| HM-GNN | 96.32 (2.61) | 86.31 (2.88) | 82.11 (7.06) |
| SAGE | 67.22 (3.68) | 53.01 (7.59) | 48.54 (10.22) |
| G2GNN | 87.23 (1.36) | 76.98 (3.47) | 71.85 (4.12) |
| N2G-GCN-VERTEX | 94.44 (2.98) | 85.89 (3.82) | 83.40 (4.58) |
| N2G-PathNNs-VERTEX | **96.78** (3.05) | **92.72** (1.11) | **88.58** (1.69) |

21.93%, 36.74%, and 21.99% on MUTAG, BZR, and COX2 respectively. Moreover, we observe a similar trend for N2G-LAMBDA (including N2G-GCN-LAMBDA, N2G-GAT-LAMBDA, and N2G-GRAPHSAGE-LAMBDA) and N2G-DELTACON (including N2G-GCN-DELTACON, N2G-GAT-DELTACON, and N2G-GRAPHSAGE-DELTACON): (i) vanilla GNN-based model suffers from performance drop under noisy labels, (ii) all N2G-based models significantly outperform vanilla model on both clean and corrupted graphs, and (iii) our N2G framework can substantially enhance the robustness of vanilla GNN-based model under noisy labels.

Moreover, Table 2 indicates that as the biased noise ratio increases, the gaps in the performances between vanilla GNN-based and N2G-based models enlarge in all cases, which confirms the premise that N2G is robust against perturbation in labels. The accuracy of the vanilla GNN-based models deteriorates significantly as the noise rate increases. In contrast, N2G-based models remain substantially more resistant to different noise levels on all three biochemical datasets. Finally, we conduct experiments on corrupted graphs using coteaching and DGNN, which are specially designed to classify data with corrupted labels. Table 13 suggests that the updating and comparison mechanism in Coteaching and the loss correction method in DGNN further enhance the N2G performance and robustness. Compared with the D-GNN method brought by [23], Table 6 shows that N2G outperforms D-GNN by 18.64% on MUTAG.

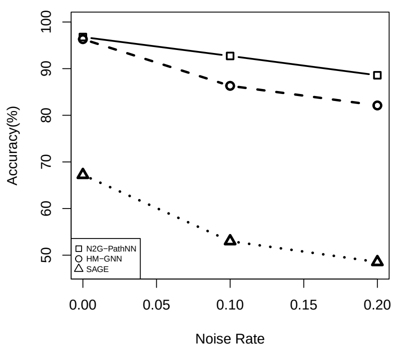


**Figure 2:** G2N vs. SOTA GoGs HM-GNN and SAGE.

**Our G2N versus other GoG approaches** To illustrate efficiency of our G2N, we compare G2N with other state-of-the-art (SOTA) GoG approaches (i.e., heterogeneous motif graph neural network (HM-GNN) [41], self-attentive graph embedding (SAGE) [18]), and G$^2$GNN [34]. For the G2N algorithm, here we consider N2G-GCN-VERTEX and N2G-PathNNs-VERTEX (i.e., we integrate N2G with the path neural networks (PathNNs) [22]). Table 5 shows that our G2N-based models can achieve competitive performances on both clean and perturbed MUTAG datasets. Specifically, we observe that N2G-PathNNs-VERTEX always outperforms all models on all datasets. Furthermore, we also compare N2G with the powerful graph of graphs neural network (PGON) [31], graph of graphs neural network (GoGNN; which is a GNN model built on the hierarchical graph structure) [30] on MUTAG. As suggested by *Table 1 in [31]*, our N2G (e.g., N2G-GCN-VERTEX with the accuracy 94.44%) significantly outperforms both PGON (accuracy is 91.15%) and GoGNN (accuracy is 88.53%).

## 4.2 N2G Results on Social Graphs

Table 3 shows classification performance on clean social networks and their counterparts with corrupted labels. Table 3 indicates that N2G outperforms all competitors on all three clean datasets. To evaluate the robustness of N2G against label perturbation, we consider a scenario of uniformly corrupted graph labels. Table 3 shows that the N2G approach outperforms all GNN-based models. Specifically, (i) When setting the ratio of noise labels of 0.1, our N2G-GRAPHSAGE-LAMBDA achieves relative gains of 9.59%, 11.49% and 11.78% over runner-ups (i.e., GRAPHSAGE) on IMDB-BINARY, IMDB-MULTI and REDDIT-BINARY, respectively, and (ii) When setting the ratio of noise labels of 0.2, our N2G-GRAPHSAGE-LAMBDA achieves relative gains of 6.55%, 9.71%

and 13.52% over runner-ups (i.e., GRAPHSAGE, GRAPHSAGE and GCN) on IMDB-BINARY, IMDB-MULTI and REDDIT-BINARY, respectively.

**Table 6:** Average accuracy (%) on MUTAG, IMDB-BINARY and IMDB-MULTI with 20% uniform corruption. DGNN results are from [23].

| **Model** | **MUTAG** | **IMDB-BINARY** | **IMDB-MULTI** |
|---|---|---|---|
| DGNN-A | 71.02 | 70.88 | 45.05 |
| DGNN-C | 57.27 | 69.40 | 47.47 |
| DGNN-E | 70.02 | 70.88 | 46.33 |
| N2G (ours) | **84.26** | **73.93** | **51.42** |

The results of our N2G-based model with biased noisy labels are summarized in Table 4. As Table 4 indicates, N2G-based models always outperform GNN-based models and achieve relative gain ranging from 86% to 22.26% in all cases on three datasets with biased corruption. Hence, we can conclude that our N2G representation substantially enhances the robustness of various vanilla GNN-based models against graph label perturbations. The results from IMDB-MULTI dataset indicate that the superiority of our N2G representation method is not limited to binary classification tasks, but also can be extended to multi-class classification problems.

Furthermore, we validate our N2G methods on corrupted social graphs using Coteaching and DGNN (regarding DGNN, we consider three variants, i.e., Conservative (DGNN-C), Anchors (DGNN-A), and Exact (DGNN-E)).

**Bioinformatics vs. Social Networks** Note that N2G achieves the highest gains in robustness in bioinformatics graphs (MUTAG, BZR, and COX2) and the social network REDDIT-BINARY, while the lowest gains are delivered on IMDB-BINARY and IMDB-MULTI. This phenomenon can be attributed to much higher density exhibited by IMDB-BINARY and IMDB-MULTI, compared to that of the bioinformatics graphs and REDDIT-BINARY (see Table 1 in the supplementary material). In this case, for graph classification tasks, we can infer that sparser networks benefit more from the integration of information of similar graphs.

**Which Graph Distance to Choose?** Our studies suggest that both the type of GNN model and the type of graphs are important factors impacting the N2G performance. For instance, we find the Vertex-Edge distance is the most promising choice for GCN and GAT across all bioinformatics and social graphs, also yielding a competitive performance for GRAPHSAGE on MUTAG and BZR. In turn, Lambda distance appears to be the winner for GRAPHSAGE on both bioinformatics and social datasets. In general, the performance of various distances in social networks appears to be more scattered than in bioinformatics graphs, which may be due to the more heterogeneous graph structures of social networks. Similarly to the results of [20], such differences among N2G based on various graph similarity measures can be attributed to the underlying network density and homophily of the resulting N2G, as well as to the structural properties that each GNN tends to focus on. Nevertheless, regardless of the graph similarity measure, all variants of N2G are more competitive, particularly, in terms of robustness than their conventional counterparts.

**N2G Versatility** Note that N2G can be combined with any node classification approach. To illustrate the N2G versatility, we now integrate N2G with the most recent SOTA GNNs for node classification (the PathNNs model) and compare it to other SOTA baselines for graph classification (i.e., GIN and persistent Weisfeiler-Lehman random walk (PWLR) [24]). As Table 7 suggests, PathNNs-N2G always outperforms both GIN and PWLR on clean and perturbed MUTAG datasets. Moreover, we find that the PathNNs-N2G-VERTEX consistently achieves the best performance on all datasets.

**Inductive Learning vs. Transductive Learning** We now assess the performance of N2G in inductive and transductive settings. That is, by the inductive setting we consider the following scenario: 1) choose a fraction $r = \{0.7, 0.8, 0.9\}$ of available graphs (all graphs in this training subset are labeled), 2) construct N2G based on the selected $r$ fraction of labeled graphs; 3) train a GNN; 4) connect $1 - r$ fraction of the remaining unlabelled graphs to N2G; 5) use the pre-trained GNN for node classification to identify labels of the remaining $1 - r$ fraction of unlabelled graphs. That is, here N2G in steps 2-3 is not allowed to integrate $1 - r$ fraction of the unlabelled training set prior to GNN training. In the transductive setting, step 2 above is changed to using all labeled and unlabelled graphs to construct a joint N2G. Figure 3 shows the performance of N2G-GCN-VERTEX and GCN in the inductive learning of MUTAG and IMDB-MULTI. We find that in all inductive splitting scenarios, N2G-GCN delivers a more competitive performance than GCN. Furthermore, clearly, N2G-GCN performs better

**Table 7:** SOTA GNN for graph classification vs. N2G with SOTA GNN for node classification on MUTAG with clean and uniformly corrupted labels. Corruption rates are 10% and 20%. The best results are in bold.

| **Model** | **MUTAG** | | |
|---|---|---|---|
| | Clean | Uniform$_{10}$ | Uniform$_{20}$ |
| GIN | 89.01 (6.02) | 77.37 (11.54) | 71.05 (7.89) |
| PWLR | 89.73 (1.01) | 82.29 (1.49) | 78.25 (2.55) |
| N2G-PathNNs-VERTEX | **96.78** (3.05) | **92.72** (1.11) | **88.58** (1.69) |
| N2G-PathNNs-LAMBDA | 95.18 (2.84) | 88.76 (0.69) | 85.70 (0.70) |
| N2G-PathNNs-DELTACON | 95.98 (1.95) | 89.96 (1.26) | 86.42 (2.03) |

in transductive than in inductive settings (see Fig. 5). However, the drop in performance is minor, indicating the overall stability of N2G. (See Appendix H for additional experiments.)

## 5 Limitations

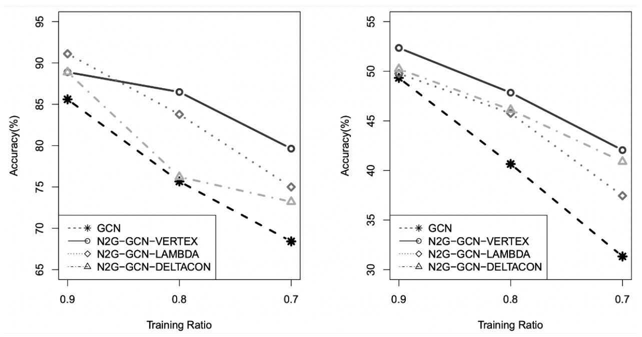


**Figure 3:** Inductive learning with N2G-GCN-VERTEX and GCN on clean graphs (*left*: MUTAG and *right*: IMDB-MULTI.). Fractions of training sets are 0.9,0.8, and 0.7.

There are two primary limitations of the current N2G approach. First, while we consider three different similarity measures among graphs, we select only one of them and, hence, may lose potentially useful information yielded by other similarity measures. This limitation can be addressed by the incorporation of an attention mechanism to automatically choose the most suitable measure. Alternatively, N2G can be extended to a multiplex N2G where all available distances are fed into the corresponding multi-graph neural networks [27]. The second limitation is that we do not incorporate the node features of individual graphs $\mathcal{G}_i$, $i = 1, 2, \ldots, N$ (wherever available). From one point of view, it implies that we use less information for graph classification tasks than conventional GNNs do, but still achieve competitive and robust performance, which is highly promising. Nevertheless, such important information as node features of individual graphs $\mathcal{G}_i$ shall not be lost. This direction can be addressed by embedding such node feature information and considering the similarity between the associated embeddings, following the ideas of [18]. While this constitutes a future research direction, we present a pilot study on these ideas in Appendix G. Finally, it is important to note that not all graph similarity measures may be uniformly feasible, as, e.g., the Vertex-Edge distance requires user-prespecified node IDs that may not be readily available in all applications.

## 6 Conclusion

We have proposed a new robust representation approach for graph learning under corrupted label scenarios, Networks to Graph (N2G). N2G reformulates the task of multiple graph classification as a node classification problem. Our results have indicated that N2G does not only improve performance but substantially enhances robustness with respect to the perturbation of graph labels. In the future, we will expand N2G to label prediction tasks for time-evolving graphs as well as advance N2G by integrating multiple graph distances simultaneously. Finally, the same idea can be applied to the classification of other objects such as images, time series, and texts, resulting in the entities to graph.

## 7 Acknowledgements

This project has been supported by NSF grants OAC-2115094, CNS-2029661, ECCS 2039701, DMS-2335846, TIP-2333703, and ONR grant N00014-21-1-2530. Also, the paper is based upon work supported by (while Y.R.G. was serving at) the NSF. The views expressed in the article do not necessarily represent the views of NSF or ONR.

### References

[1] John Adrian Bondy. *Graph theory with applications*. 1982.

[2] Wuxinlin Cheng, Chenhui Deng, Zhiqiang Zhao, Yaohui Cai, Zhiru Zhang, and Zhuo Feng. Spade: A spectral method for black-box adversarial robustness evaluation. In *ICML*, pages 1814–1824, 2021.

[3] Gregorio D'Agostino and Antonio Scala. *Networks of networks: the last frontier of complexity*, volume 340. Springer, 2014.

[4] Enyan Dai, Charu Aggarwal, and Suhang Wang. NRGNN: learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao, editors, *SIGKDD*, pages 227–236, 2021.

[5] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *WSDM*, pages 181–191, 2022.

[6] Siemon C de Lange, Marcel A de Reus, and Martijn P van den Heuvel. The laplacian spectrum of neural networks. *Frontiers in computational neuroscience*, 7:189, 2014.

[7] Xuefeng Du, Tian Bian, Yu Rong, Bo Han, Tongliang Liu, Tingyang Xu, Wenbing Huang, Yixuan Li, and Junzhou Huang. Noise-robust graph learning by estimating and leveraging pairwise interactions. *Transactions on Machine Learning Research*, 2023.

[8] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[9] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*. MIT press, 2007.

[10] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *NIPS*, 30, 2017.

[12] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. In *ICML*, pages 8230–8248, 2022.

[13] Shonosuke Harada, Hirotaka Akita, Masashi Tsubaki, Yukino Baba, Ichigaku Takigawa, Yoshihiro Yamanishi, and Hisashi Kashima. Dual graph convolutional neural network for predicting chemical networks. *BMC Bioinformatics*, 21:1–13, 2020.

[14] Manfred Jaeger. Learning and reasoning with graph data: Neural and statistical-relational approaches. In *AiB*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[15] Hassan Khosravi and Bahareh Bina. A survey on statistical relational learning. In *Advances in Artificial Intelligence: 23rd Canadian Conference on Artificial Intelligence*, pages 256–268, 2010.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[17] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. Deltacon: A principled massive-graph similarity function. In *SDM*, pages 162–170. SIAM, 2013.

[18] Jia Li, Yu Rong, Hong Cheng, Helen Meng, Wenbing Huang, and Junzhou Huang. Semi-supervised graph classification: A hierarchical graph perspective. In *The World Wide Web Conference*, pages 972–982, 2019.

[19] Yuwen Li, Miao Xiong, and Bryan Hooi. Graphcleaner: Detecting mislabelled samples in popular graph learning benchmarks. In *ICML*, 2023.

[20] Pengqian Lu. *A Homophilous and Dense Graph is Robust to Label Noise*. PhD thesis, The University of Sydney, 2022.

[21] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. Dimensionality-driven learning with noisy labels. In *ICML*, pages 3355–3364, 2018.

[22] Gaspard Michel, Giannis Nikolentzos, Johannes F Lutzeyer, and Michalis Vazirgiannis. Path neural networks: Expressive and accurate graph neural networks. In *International Conference on Machine Learning*, pages 24737–24755. PMLR, 2023.

[23] Hoang NT, Choong Jun Jin, and Tsuyoshi Murata. Learning graph neural networks with noisy labels. *arXiv:1905.01591*, 2019.

[24] Sun Woo Park, Yun Young Choi, Dosang Joe, U Jin Choi, and Youngho Woo. The pwlr graph representation: A persistent weisfeiler-lehman scheme with random walks for graph classification. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pages 287–297. PMLR, 2022.

[25] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *IEEE CVPR*, pages 1944–1952, 2017.

[26] Siyi Qian, Haochao Ying, Renjun Hu, Jingbo Zhou, Jintai Chen, Danny Z Chen, and Jian Wu. Robust training of graph neural networks via noise governance. In *WSDM*, pages 607–615, 2023.

[27] Josephine M Thomas, Alice Moallemy-Oureh, Silvia Beddar-Wiesing, and Clara Holzhüter. Graph neural networks designed for different graph types: A survey. *Transactions on Machine Learning Research*, 2023.

[28] Petar Veličković. Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538, 2023.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

[30] Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, and Xuemin Lin. Gognn: Graph of graphs neural network for predicting structured entity interactions. *arXiv preprint arXiv:2005.05537*, 2020.

[31] Hanchen Wang, Defu Lian, Wanqi Liu, Dong Wen, Chen Chen, and Xiaoyang Wang. Powerful graph of graphs neural network for structured entity analysis. *World Wide Web*, 25(2):609–629, 2022.

[32] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. AM-GCN: Adaptive multi-channel graph convolutional networks. In *SIGKDD*, pages 1243–1253, 2020.

[33] Yizhen Wang, Somesh Jha, and Kamalika Chaudhuri. Analyzing the robustness of nearest neighbors to adversarial examples. In *ICML*, pages 5133–5142, 2018.

[34] Yu Wang, Yuying Zhao, Neil Shah, and Tyler Derr. Imbalanced graph classification via graph-of-graph neural networks. In *CIKM*, pages 2067–2076, 2022.

[35] Peter Wills and François G. Meyer. Metrics for graph comparison: A practitioner's guide. *PLOS ONE*, 15(2):e0228728, 2020. doi: 10.1371/journal.pone.0228728. URL https://app.dimensions.ai/details/publication/pub.1124845369.

[36] Richard C Wilson and Ping Zhu. A study of graph spectra for comparing graphs and trees. *Pattern Recognition*, 41(9):2833–2841, 2008.

[37] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer, 2022.

[38] 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.

[39] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[40] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *ICML*, pages 7164–7173, 2019.

[41] Zhaoning Yu and Hongyang Gao. Molecular representation learning via heterogeneous motif graph neural networks. In *ICML*, pages 25581–25594, 2022.

[42] Jingyang Yuan, Xiao Luo, Yifang Qin, Yusheng Zhao, Wei Ju, and Ming Zhang. Learning on graphs under label noise. In *ICASSP*, pages 1–5. IEEE, 2023.

[43] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115, 2021.

[44] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

## A Graph Similarity Measures

To define weighted edges $e_{ij}$, we consider three graph similarity measures as possible choices for $\mathscr{D}(\cdot,\cdot)$ (here the lower the distance is, the more similar the two graphs are):

**Vertex-Edge distance** is the inversion of the vertex-edge overlap, which is formulated with:

$$\mathscr{D}^{\text{VERTEX}}(\mathcal{G}_i,\mathcal{G}_j)) = \frac{(1 - VEO(\mathcal{G}_i,\mathcal{G}_j))}{VEO(\mathcal{G}_i,\mathcal{G}_j)},$$

where the vertex-edge overlap is defined as:

$$VEO(\mathcal{G}_i,\mathcal{G}_j)) = \frac{|V_i \cap V_j| + |E_i \cap E_j|}{|V_i| + |V_j| + |E_i| + |E_j|}.$$

Here $|V_i \cap V_j|$ and $|E_i \cap E_j|$ are the common vertices and edges between graphs $\mathcal{G}_i$ and $\mathcal{G}_j$, respectively. As such, the Vertex-Edge distance measures the graph similarity based on their vertex-edge overlap (for more details see [8]).

**Remark.** The important limitations of the Vertex-Edge distance is that it requires pre-specified node IDs that typically come from the expert knowledge. However, it is important to note that such node IDs may not be available in all applications or may require resource-intensive labelling process. As an alternative, we can use the modified Vertex-Edge distance where the overlap (intersection of sets) is substituted by a union. Our numerical results suggest that using such modified Vertex-Edge distance still result in competitive performance.

**Lambda distance** is the Euclidean norm of the difference among the eigenvalues of the normalized graph Laplacians of $\mathcal{G}_i$ and $\mathcal{G}_j$:

$$\mathscr{D}^{\text{LAMBDA}}(\mathcal{G}_i,\mathcal{G}_j)) = ||L_i^s - L_j^s||,$$

where $L_i^s$ and $L_j^s$ are the top $s$ eigenvalues of the normalized graph Laplacian matrices of graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ (for more details see [36]).

**DeltaCon distance** [17] is the Matusita difference of the fast belief propagation matrix, which weights the information from neighbors with parameter $\epsilon \in (0, 1)$:

$$\mathscr{D}^{\text{DELTACON}}(\mathcal{G}_i,\mathcal{G}_j) = \sqrt{\sum_{n,m}\left(\sqrt{s^i_{n,m}} - \sqrt{s^j_{n,m}}\right)^2}.$$

Here $s^i_{n,m}$ is the element in the faster belief propagation matrix $S_i = \left[I + \epsilon^2 D_i - \epsilon A_i\right]^{-1}$, where $A_i$ and $D_i$ are the adjacency and degree matrices of graph $\mathcal{G}_i$. (For the more detailed discussion of graph distances see [35].)

**Algorithm 1** N2G Algorithm

```
1: Input: Graphs $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$, distance function $\mathscr{D}(\cdot,\cdot)$[1], threshold $\tau$[2]
2: Output: the final N2G graph $\mathbb{G}_\tau$
3: for $\mathcal{G}_i = 1, 2, \ldots, N-1$ do
4:     for $\mathcal{G}_j = i+1, i+2, \ldots, N$ do
5:         Compute $\mathscr{D}(\cdot,\cdot)$ between $\mathcal{G}_i$ and $\mathcal{G}_j$
6:         Form the edge $e_{ij} = \mathscr{D}(\mathcal{G}_i,\mathcal{G}_j)$, if $\mathscr{D}(\mathcal{G}_i,\mathcal{G}_j) \le \tau$ and 0, otherwise
7:     end for
8: end for
9: Construct a new graph-structured object $\mathbb{G}$ with edges $\{e_{ij}\}_{ij}$ and nodes $\mathcal{G}_i$, $i = 1, 2, \ldots, N$.
10: Feed $\mathbb{G}$ into any type of GNN-based model.
```

[1] $\mathscr{D}(\cdot,\cdot)$ can be $\mathscr{D}^{\text{VERTEX}}(\cdot,\cdot)$, $\mathscr{D}^{\text{LAMBDA}}(\cdot,\cdot)$, $\mathscr{D}^{\text{DELTACON}}(\cdot,\cdot)$ [2]Threshold $\tau$ can be selected using crossvalidation.

## B Datasets and Experimental Details

**Hyperparameters** Similar to [16], We use $(\ell + 2)$-layers learning structures consisting of 1 input layer, $\ell$ hidden layer (where $\ell \in \{3, 4, 5, 6\}$), and 1 output layer for all models. For search area, we set the learning rate$(lr) \in \{0.1, 0.01, 0.001\}$, hidden units $(nhid) \in \{32, 64, 128, 256, 512\}$, the threshold of N2G construction $\tau \in \{0.25, 0.5, 0.75\}$ for MUTAG, BZR, and COX2 datasets, and $\tau \in \{0.05, 0.1, 0.25, 0.5, 0.75\}$ for IMDB-BINARY, IMDB-MULTI, and REDDIT-BINARY datasets, and the batch size $\in \{24, 32, 64, 128\}$. For optimization, we use Adam optimizer and set the number of epochs ranging from 500 to 1500 depending on the convergence rate.

**Baselines** The proposed N2G is compared or integrated with the following state-of-the-art models:

- Graph Convolutional Network (GCN) [16]: GCN is a semi-supervised graph convolutional network.
- Graph Isomorphism Network (GIN) [39]: GIN is the graph neural network model which is as expressive as the Weisfeiler-Lehman graph isomorphism test.
- Graph Attention Networks (GAT) [29]: GAT leverages the attention mechanism for the homogeneous graph which includes only one type of nodes or links.
- GraphSAGE [11]: GraphSAGE learns node representations through aggregation of neighborhood information.
- Adaptive Multi-channel Graph Convolutional Networks (AM-GCN) [32]: AM-GCN is the graph neural network model which uses the attention mechanism to learn adaptive importance weights of the embeddings.
- Denoising Graph Neural Network (DGNN) [23]: DGNN takes noise-correction approach (i.e., a noise estimator) to train a graph neural network with noisy labels.
- Persistent Weisfeiler-Lehman Random Walk (PWLR) [24]: PWLR is a mathematical framework which produces a collection of explainable low-dimensional representations of graphs with discrete and continuous node features.
- Heterogeneous Motif Graph Neural Network (HM-GNN) [41]: HM-GNN learns node feature representations for each node in the heterogeneous motif graph.
- Self-Attentive Graph Embedding (SAGE) [18]: SAGE embeds graph instances of arbitrary size into fixed-length vectors.
- Path Neural Networks (PathNNs) [22]: PathNNs updates node representations by aggregating paths emanating from nodes.
- G-Mixup [12]: G-Mixup augments graphs for graph classification by interpolating the generator (i.e., graphon) of different classes of graphs. In G-Mixup, it uses GCN/GIN as the base architecture.
- G$^2$GNN [34]: G$^2$GNN alleviates the graph imbalance issue by deriving extra supervision globally from neighboring graphs and locally from stochastic augmentations of graphs.

**Table 9:** Dataset Summaries.

| **Dataset** | **# Graphs** | **Avg. $\vert\mathcal{V}\vert$** | **Avg. $\vert\mathcal{E}\vert$** | **Avg. Density** | **# Class** |
|---|---|---|---|---|---|
| MUTAG | 188 | 17.93 | 19.79 | 0.1304 | 2 |
| BZR | 405 | 35.75 | 38.35 | 0.0617 | 2 |
| COX2 | 467 | 41.22 | 43.45 | 0.0524 | 2 |
| IMDB-BINARY | 1000 | 19.77 | 96.53 | 0.5203 | 2 |
| IMDB-MULTI | 1500 | 13.00 | 65.94 | 0.8454 | 3 |
| REDDIT-BINARY | 2000 | 429.63 | 497.75 | 0.0005 | 2 |

## C How Does N2G Work with the Additional GNN Architecture?

As we noted before, our primary goal is to evaluate the utility of N2G as a general tool that can be coupled with **any** GNNs, rather than to promote a particular GNN model. In the main body, we illustrate the integration of N2G with three (arguably) most popular vanilla GNN architectures (GCN, GAT, GRAPHSAGE) as well as with one of the most currently competitive GNN models for node classification PathNNs. Here we provide additional results on PathNNs-N2G on BZR and COX2 (see Table 10). Similar as in the case of MUTAG (see Table 7), N2G tends to yield the competitive performance both in terms of classification accuracy and robustness across all considered scenarios. The best results are achieved by N2G with vertex similarity.

**Table 10:** SOTA GNN for graph classification vs. N2G with SOTA GNN for node classification on BZR, COX2 and their counterparts under uniform corruption of graph labels. Corruption rates are 10% and 20%. Standard deviation is in ( ). The best results are in bold.

| **Model** | **BZR** | | | **COX2** | | |
|---|---|---|---|---|---|---|
| | Clean | $\text{Uniform}_{10}$ | $\text{Uniform}_{20}$ | Clean | $\text{Uniform}_{10}$ | $\text{Uniform}_{20}$ |
| GIN | 85.63 (2.59) | 73.49 (3.79) | 68.93 (4.15) | 83.51 (4.75) | 73.49 (3.79) | 68.93 (4.15) |
| PWLR | 85.46 (0.55) | 79.31 (1.57) | 72.74 (0.80) | 79.94 (0.58) | 73.48 (1.79) | 69.14 (3.38) |
| N2G-PathNN-VERTEX | **90.39** (2.74) | **88.94** (1.18) | **85.18** (0.98) | **90.41** (4.02) | **88.19** (0.54) | **85.55** (0.94) |
| N2G-PathNN-LAMBDA | 88.96 (3.19) | 85.12 (4.12) | 82.61 (0.99) | 88.02 (0.55) | 84.50 (0.20) | 83.77 (0.65) |
| N2G-PathNN-DELTACON | 88.16 (3.45) | 86.77 (1.69) | 83.07 (1.07) | 88.89 (1.18) | 86.95 (2.03) | 83.48 (1.98) |

We also present a study on the utility of our N2G approach when it is combined with another powerful GNN model – the AM-GCN [32]. In particular, we select AM-GCN due to its capability to adaptively learn deep correlation information from both topological information and node features. Table 11 indicates that compared with AM-GCN[1], N2G coupled with AM-GCN delivers consistently competitive results on both clean and corrupted MUTAG, thereby reconfirming the important role of the essential relational information and broader utility of N2G.

**Table 11:** Performance of N2G coupled with AM-GCN on on clean and uniformly corrupted MUTAG. Corruption rates are 10% and 20%. Standard deviation is in ( ). Corruption rates are 10% and 20%. Standard deviation is in ( ). Best results are in bold.

| **Model** | **MUTAG** | | |
|---|---|---|---|
| | Clean | Uniform10 | Uniform20 |
| AM-GCN | 85.73 (7.38) | 80.16 (6.46) | 74.28 (4.98) |
| N2G-AM-GCN-VERTEX | **92.76** (2.78) | **85.31** (1.87) | **81.30** (3.45) |
| N2G-AM-GCN-LAMBDA | 90.39 (3.02) | 84.72 (2.88) | 79.10 (1.95) |
| N2G-AM-GCN-DELTACON | 90.71 (1.87) | 84.02 (3.47) | 80.10 (2.11) |

## D Sensitivity to the Threshold Selection

In our experiments, we select threshold $\tau$ for the edge construction from the quantile set of the empirical distribution of the graph distances $\{\mathscr{D}_{ij}\}_{i,j=1}^{N}$, via the cross-validation. To evaluate sensitivity of N2G to the choice of $\tau$, we consider performance of N2G-GCN on clean MUTAG and its counterparts under uniform corruption of graph labels with rates of 10% and 20%. Here $\tau$ undertakes values from the quartile set (i.e., 25%, 50%, and 75%). As Table 12 shows, the N2G performance tends to be stable across various selections of $\tau$ and graph distances, with highly competitive results corresponding to lower values of $\tau$ for larger graphs (i.e., larger graphs tend to result in sparser $\mathbb{G}$). This phenomenon may be potentially attributed to the fact that for larger individual graphs dependence among graphs plays a less significant role, and N2G inherently tends to focus on the most essential similarities and prunes less important ones.

[1]Note that, for vanilla AM-GCN, we consider the original adjacency matrix and latent node embedding as input.

**Table 12:** Sensitivity of performance with respect to threshold $\tau$. Average accuracy on clean bioinformatics graphs and their counterparts under uniform corruption of graph labels. Corruption rates are 10% and 20%. Standard deviation is in ( ).

| Model | **MUTAG** $\tau = 25\%$ Clean | Uniform$_{10}$ | Uniform$_{20}$ | **MUTAG** $\tau = 50\%$ Clean | Uniform$_{10}$ | Uniform$_{20}$ | **MUTAG** $\tau = 75\%$ Clean | Uniform$_{10}$ | Uniform$_{20}$ |
|---|---|---|---|---|---|---|---|---|---|
| GCN | 85.60 (5.80) | 70.88 (2.99) | 68.60 (4.02) | 85.60 (5.80) | 70.88 (2.99) | 68.60 (4.02) | 85.60 (5.80) | 70.88 (2.99) | 68.60 (4.02) |
| N2G-GCN-VERTEX | **94.44** (2.98) | **85.89** (3.82) | **83.40** (4.58) | **92.12** (2.09) | **84.16** (4.54) | **81.90** (3.40) | **89.47** (3.41) | **82.78** (4.07) | **77.72** (1.95) |
| N2G-GCN-LAMBDA | 92.58 (4.85) | 82.81 (3.55) | 81.41 (3.08) | 90.17 (1.73) | 81.44 (2.69) | 79.14 (2.76) | 87.07 (1.69) | 80.14 (2.60) | 77.08 (3.48) |
| N2G-GCN-DELTACON | 89.19 (2.88) | 80.17 (3.14) | 78.64 (2.48) | 87.49 (3.57) | 82.23 (3.19) | 78.85 (3.22) | 86.16 (2.96) | 78.42 (2.01) | 75.43 (3.72) |

## E N2G with Coteaching and Loss Correction

In addition, to resist noisy labels, we have applied the coteaching and loss correction to our N2G model. Since D-GNN [23] involves loss correction and we use the same approach to corrupt graph labels, we consider here integration of N2G with D-GNN [23].

Table 14 indicates that coteaching and the loss correction method in DGNN incorporated with N2G method can further improve the classification performance over different types of label corruption. In particular, N2G-DGNN outperforms DGNN on both IMDB-BINARY and IMDB-MULTI datasets under all scenarios of label corruption.

**Table 13:** N2G with coteaching on bioinformatics graphs with noisy graph labels. Label corruptions are either uniform or biased, with rates of 10% and 20%. Standard deviation is in ( ). The best results are in bold.

| **Model** | **MUTAG** Uniform$_{10}$ | Uniform$_{20}$ | Biased$_{10}$ | Biased$_{20}$ | **BZR** Uniform$_{10}$ | Uniform$_{20}$ | Biased$_{10}$ | Biased$_{20}$ | **COX2** Uniform$_{10}$ | Uniform$_{20}$ | Biased$_{10}$ | Biased$_{20}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N2G-Coteaching-VERTEX | **87.42** (2.36) | **85.42** (3.67) | **79.29** (4.56) | **76.28** (4.35) | **86.91** (2.12) | **84.79** (3.42) | **81.19** (4.30) | **77.10** (3.45) | **83.82** (3.12) | **81.12** (2.80) | **80.27** (4.19) | **78.82** (2.51) |
| N2G-Coteaching-LAMBDA | 83.45 (2.89) | 82.34 (3.70) | 78.57 (3.01) | 74.45 (4.80) | 84.67 (3.95) | 83.62 (1.93) | 81.26 (3.68) | 78.35 (4.22) | 83.28 (4.20) | 81.16 (3.65) | 79.41 (3.29) | 76.95 (3.57) |
| N2G-Coteaching-DELTACON | 82.48 (4.96) | 79.89 (3.67) | 78.99 (2.79) | 72.34 (3.88) | 85.94 (2.59) | 84.58 (4.75) | 80.45 (3.69) | 79.54 (2.93) | 83.71 (2.85) | 80.62 (3.56) | 80.24 (2.97) | 78.05 (3.74) |
| N2G-DGNN-VERTEX | **86.35** (2.88) | **84.26** (2.93) | **79.11** (5.07) | **77.82** (2.51) | **83.17** (4.85) | **82.55** (3.79) | **79.50** (3.91) | **77.82** (2.51) | **81.90** (2.78) | **80.89** (4.75) | **80.27** (4.19) | **77.82** (2.51) |
| N2G-DGNN-LAMBDA | 83.38 (4.17) | 82.26 (3.13) | 78.24 (1.35) | 76.95 (3.57) | 81.79 (3.50) | 80.98 (4.08) | 79.11 (2.82) | 76.95 (3.57) | 81.29 (3.25) | 80.05 (3.22) | 79.41 (3.29) | 76.95 (3.57) |
| N2G-DGNN-DELTANCON | 83.14 (4.06) | 81.04 (3.77) | 79.05 (3.46) | 77.06 (3.74) | 82.96 (4.91) | 81.77 (3.60) | 77.82 (3.80) | 77.05 (3.74) | 81.46 (2.85) | 80.62 (3.56) | 80.24 (2.97) | 77.62 (3.56) |

**Table 14:** N2G with coteaching on social graphs with noisy graph labels. Label corruptions are either uniform or biased, with rates of 10% and 20%. Standard deviation is in ( ). The best results are in bold.

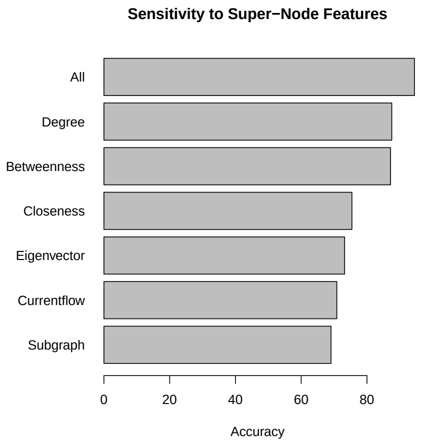
| **Model** | **IMDB-BINARY** Uniform$_{10}$ | Uniform$_{20}$ | Biased$_{10}$ | Biased$_{20}$ | **IMDB-MULTI** Uniform$_{10}$ | Uniform$_{20}$ | Biased$_{10}$ | Biased$_{20}$ | **REDDIT-BINARY** Uniform$_{10}$ | Uniform$_{20}$ | Biased$_{10}$ | Biased$_{20}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N2G-Coteaching-VERTEX | **76.38** (4.48) | **73.93** (5.04) | **75.84** (4.73) | **72.14** (3.84) | **53.41** (3.78) | **51.42** (5.78) | **53.17** (4.01) | **50.90** (3.83) | **82.34** (5.53) | **80.14** (4.77) | **80.47** (5.07) | **77.41** (4.45) |
| N2G-Coteaching-LAMBDA | 75.21 (4.05) | 72.44 (3.99) | 73.89 (4.82) | 71.33 (5.66) | 52.15 (5.00) | 50.11 (1.92) | 48.75 (3.04) | 46.33 (4.60) | 80.41 (4.04) | 77.26 (5.16) | 77.32 (2.24) | 76.08 (3.41) |
| N2G-Coteaching-DELTACON | 75.05 (5.53) | 72.60 (4.63) | 70.85 (4.76) | 68.67 (2.95) | 52.41 (3.77) | 49.04 (4.05) | 49.22 (4.18) | 46.03 (3.76) | 78.74 (4.53) | 76.01 (2.87) | 76.85 (4.27) | 75.12 (4.75) |
| N2G-DGNN-VERTEX | **76.14** (3.90) | **73.42** (5.17) | **73.88** (4.02) | **71.12** (4.23) | **53.14** (5.23) | **50.51** (4.90) | **52.03** (1.80) | **49.11** (5.43) | **83.85** (3.93) | **80.24** (5.10) | **77.74** (5.26) | **76.12** (2.04) |
| N2G-DGNN-LAMBDA | 74.12 (4.55) | 72.10 (3.53) | 71.87 (3.96) | 66.54 (3.24) | 48.44 (3.86) | 45.86 (2.98) | 51.42 (4.70) | 49.04 (2.15) | 80.48 (3.25) | 77.49 (4.66) | 76.87 (5.01) | 74.01 (3.25) |
| N2G-DGNN-DELTACON | 76.47 (4.77) | 72.88 (6.04) | 70.14 (2.49) | 67.20 (2.88) | 50.84 (3.46) | 47.64 (3.28) | 49.81 (3.71) | 46.01 (3.74) | 80.23 (5.10) | 78.94 (4.44) | 76.18 (5.22) | 75.01 (3.21) |

## F Ablation Study on Super-Node Features

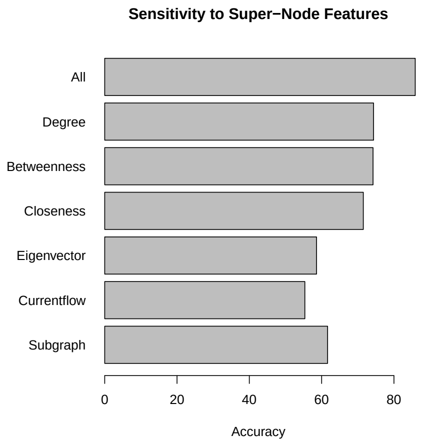
Each super-node of N2G (i.e., graph $\mathcal{G}_i$, $i = 1, 2, \dots, N$) is associated with features which are various network statistics. We consider average degree centrality, betweenness centrality, closeness centrality, eigenvector centrality, current flow betweenness centrality, subgraph centrality, and current flow closeness centrality. Figure 4 presents an ablation study on the contribution of average degree centrality and betweenness centrality for N2G with GCN over various graph distances for clean and perturbed MUTAG. (Results for other network statistics and models are analogous and omitted for brevity.) Figure 4 indicates the following:

- **Accuracy** Across all scenarios, MUTAG N2G with all super-node features yields more accurate results than N2G with solely degree centrality or betweenness centrality. The gains of N2G with all super-node features vs. its counterparts solely feature range from 2% to 36.77% on clean graphs.
- **Robustness** It appears that using all super-node features is most beneficial in a case of perturbed graphs. The gains of N2G with all super-node features vs. N2G with solely feature is up to 55.15% for the perturbation rate of 20%, and generally, the robustness gains of N2G with all super-node features are higher for higher perturbation rates. These phenomena can be explained by the complementary strength each super-node feature brings to the learning process.

- Finally, contributions of degree centrality and betweenness centrality for clean and perturbed MUTAG are generally comparable.



**(a)** Clean MUTAG



**(b)** Noisy MUTAG with 10 percent noise

**Figure 4:** N2G-GCN-VERTEX sensitivity to super node features.

**Table 15:** Transductive and Inductive Learning on clean IMDB-MULTI with training ratios $r = 0.9, 0.8, 0.7$. Standard deviation is in ( ).

| **Model** | **IMDB-MULTI-9-1** | | **IMDB-MULTI-8-2** | | **IMDB-MULTI-7-3** | |
|---|---|---|---|---|---|---|
| | Transductive | Inductive | Transductive | Inductive | Transductive | Inductive |
| GCN | N/A | 49.33 (3.21) | N/A | 40.65 (2.92) | N/A | 31.33 (0.01) |
| N2G-GCN-VERTEX | **56.81** (5.05) | **52.35** (2.56) | **53.49** (3.25) | **47.84** (3.12) | **47.45** (5.12) | **42.05** (4.88) |
| N2G-GCN-LAMBDA | 51.81 (5.05) | 49.77 (2.85) | 48.56 (2.99) | 45.76 (4.42) | 43.11 (3.74) | 37.45 (4.23) |
| N2G-GCN-DELTACON | 52.74 (3.47) | 50.23 (1.99) | 49.16 (1.03) | 46.12 (2.52) | 44.90 (2.13) | 40.89 (5.08) |

**Table 16:** Transductive and Inductive Learning on clean MUTAG with training ratios $r = 0.9, 0.8, 0.7$.

| **Model** | **MUTAG-9-1** | | **MUTAG-8-2** | | **MUTAG-7-3** | |
|---|---|---|---|---|---|---|
| | Transductive | Inductive | Transductive | Inductive | Transductive | Inductive |
| GCN | N/A | 85.60 (5.80) | N/A | 75.67 (1.95) | N/A | 68.42 (3.10) |
| N2G-GCN-VERTEX | **94.44** (2.98) | 88.89 (0.01) | **91.33** (4.85) | **86.49** (0.01) | **88.89** (2.49) | **79.64** (3.50) |
| N2G-GCN-LAMBDA | 92.58 (4.85) | **91.11** (2.72) | 87.43 (3.44) | 83.78 (0.01) | 87.13 (3.01) | 75.00 (0.00) |
| N2G-GCN-DELTACON | 89.19 (2.88) | 88.89 (0.01) | 84.13 (3.01) | 76.22 (1.08) | 80.29 (2.82) | 73.21 (0.01) |

## G Pilot Study on Using Node Features of Individual Graphs

We present a pilot study on the utility of node features of the individual graphs $\mathcal{G}_i, i = 1, 2, \ldots, N$ in the N2G learning framework. We consider two bioinformatics graphs BZR and COX2, each with three node attributes. We take the maximum of each of the three node features over each $\mathcal{G}_i$ and associate the resulting three node summaries with the corresponding $\mathcal{G}_i$, $i = 1, 2, \ldots, N$. Hence, coupled with the seven network summary statistics, each super-node in N2G (i.e.,$\mathcal{G}_i$) is now associated with ten node features. Table 17 presents the obtained results for N2G coupled with GCN. In general, we find that integrating individual node features into N2G results in improving classification accuracy both on clean and perturbed graphs. However, in most cases the gains are minor, with N2G-N2G-VERTEX-WITH NODE FEATURES being the leader. Nevertheless, we view these findings as a promising indication that node features of $\mathcal{G}_i$ have a potential to further enhance GoG capabilities, and we will explore this direction in our future work.

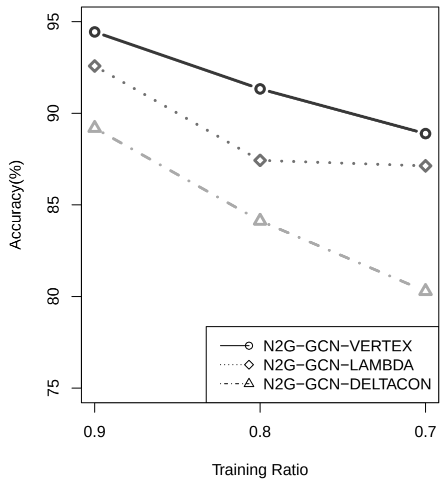
**Table 17:** Average graph classification accuracy with and without node features of individual bioinformatics graphs. Uniform corruption rates are 10% and 20%. Standard deviation is in ( ). The best results are in bold.

| **Model** | **BZR** | | | **COX2** | | |
|---|---|---|---|---|---|---|
| | Clean | Uniform$_{10}$ | Uniform$_{20}$ | Clean | Uniform$_{10}$ | Uniform$_{20}$ |
| GCN | 80.49 (3.22) | 67.36 (3.74) | 60.91 (3.25) | 78.60 (1.52) | 69.09 (5.28) | 65.72 (6.25) |
| N2G-GCN-VERTEX | 87.22 (3.71) | 84.36 (4.12) | 83.29 (3.10) | 86.74 (4.63) | 82.44 (4.07) | 80.17(3.09) |
| N2G-GCN-VERTEX-WITH NODE FEATURES | **88.19** (3.14) | **84.96** (2.74) | **84.14** (3.18) | **86.95** (2.03) | **83.82** (1.89) | **81.47**(4.02) |
| N2G-GCN-LAMBDA | 85.19 (1.93) | 81.76 (3.30) | 80.94 (2.66) | 84.07 (1.69) | 80.74 (3.99) | 78.01 (4.94) |
| N2G-GCN-LAMBDA-WITH NODE FEATURES | 85.70 (2.01) | 82.92 (1.09) | 81.34 (4.06) | 84.77 (3.19) | 82.40 (2.90) | 79.85 (3.17) |
| N2G-GCN-DELTACON | 86.50 (3.96) | 81.55 (4.67) | 80.44 (2.37) | 84.26 (3.66) | 79.85 (4.91) | 78.63 (4.17) |
| N2G-GCN-DELTACON-WITH NODE FEATURES | 86.50 (3.96) | 81.55 (4.67) | 80.44 (2.37) | 84.26 (3.66) | 79.85 (4.91) | 78.63 (4.17) |

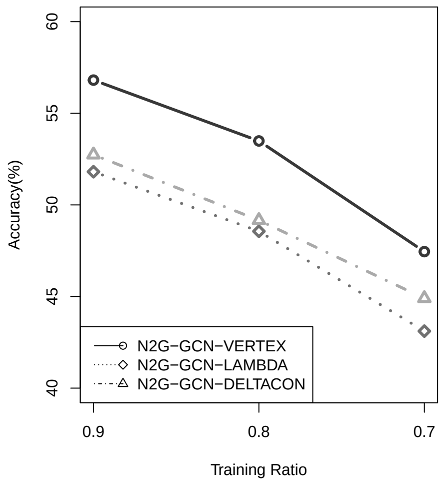
## H Inductive vs. Transductive Learning

Fig. 5 shows the results of transductive learning on MUTAG with N2G-GCN-VERTEX and GCN, where fractions $r$ of training sets are 0.7, 0.8, and 0.9. By the inductive setting we consider the following scenario: 1) choose a fraction $r = \{0.7, 0.8, 0.9\}$ of available graphs (all graphs in this training subset are labelled), 2) construct N2G based on all available labelled and unlabelled graphs; 3) train a GNN; 5) use the resulting GNN for node classification to identify labels of the remaining $1 - r$ fraction of unlabelled graphs. Note that under this scenario we do not compare with the baseline GCN for graph classification as such baseline GCN for graph classification by default always performs in the inductive regime. Fig. 5 suggests that N2G based on vertex similarity consistently delivers the most competitive performance, even when the fraction of the training set reduces to 70%.

In turn, Table 15 presents the experiments on inductive and transductive learning on IMDB-MULTI. The obtained results echo the findings obtained for inductive and transductive learning on MUTAG (see Figs. 3 and 5 as well as their Table counterpart 16) and suggest the overall stability and consistency of the N2G performance.



**(a)** MUTAG.



**(b)** IMDB-MULTI.

**Figure 5:** Transductive learning with N2G-GCN-VERTEX and GCN. Fractions $r$ of training sets are 0.9, 0.8, and 0.7.

## I Notation

Table 18 includes the primary notations used in this paper.

**Table 18:** Table of the key notations.

| | |
|---|---|
| $A$: | Adjacency matrix |
| $D$: | Degree matrix |
| $\mathscr{D}_{ij} = \mathscr{D}(\mathcal{G}_i, \mathcal{G}_j)$: | Similarity measure between graphs $\mathcal{G}_i$ and $\mathcal{G}_j$, $i, j = 1, 2, \ldots, N$ |
| $e_{ij}$: | Edge between between graphs $\mathcal{G}_i$ and $\mathcal{G}_j$, $i, j = 1, 2, \ldots, N$ |
| $\tilde{\mathcal{G}}$: | Set of observed graphs |
| $\mathcal{G}_i$: | Graph $\mathcal{G}_i$ in $\tilde{\mathcal{G}}$, $i = 1, 2, \ldots, N$ |
| $\mathbb{G}$: | Networks to Graph (N2G) |
| $L_i^s$: | top $s$ eigenvalues of the normalized Laplacian of graph $\mathcal{G}_i$ |
| $N$: | Number of graphs in $\tilde{\mathcal{G}}$ |
| $S$: | Faster belief propagation matrix |
| $\tau$: | The threshold for N2G |