Breaking the Limit of Graph Neural Networks by Improving the Assortativity of Graphs with Local Mixing Patterns

Susheel Suresh
Department of Computer Science
Purdue University
West Lafayette, Indiana, USA
suresh43@purdue.edu

Vinith Budde
Alexa AI
Amazon
Seattle, Washington, USA
buddev@amazon.com

Jennifer Neville
Department of Computer Science
Purdue University
West Lafayette, Indiana, USA
neville@purdue.edu

Pan Li
Department of Computer Science
Purdue University
West Lafayette, Indiana, USA
panli@purdue.edu

Jianzhu Ma
Department of Computer Science
Purdue University
West Lafayette, Indiana, USA
ma634@purdue.edu

ABSTRACT

Graph neural networks (GNNs) have achieved tremendous success on multiple graph-based learning tasks by fusing network structure and node features. Modern GNN models are built upon iterative aggregation of neighbor's/proximity features by message passing. Its prediction performance has been shown to be strongly bounded by *assortative mixing* in the graph, a key property wherein nodes with similar attributes mix/connect with each other. We observe that real world networks exhibit heterogeneous or diverse mixing patterns and the conventional global measurement of assortativity, such as global assortativity coefficient, may not be a representative statistic in quantifying this mixing. We adopt a generalized concept, node-level assortativity, one that is based at the node level to better represent the diverse patterns and accurately quantify the learnability of GNNs. We find that the prediction performance of a wide range of GNN models is highly correlated with the node level assortativity. To break this limit, in this work, we focus on transforming the input graph into a computation graph which contains both proximity and structural information as distinct type of edges. The resulted multi-relational graph has an enhanced level of assortativity and, more importantly, preserves rich information from the original graph. We then propose to run GNNs on this computation graph and show that adaptively choosing between structure and proximity leads to improved performance under diverse mixing. Empirically, we show the benefits of adopting our transformation framework for semi-supervised node classification task on a variety of real world graph learning benchmarks.

CCS CONCEPTS

• **Computing methodologies → Statistical relational learning**; *Knowledge representation and reasoning*; *Network science*.

KEYWORDS

graph neural networks; mixing patterns; representation learning

ACM Reference Format:
Susheel Suresh, Vinith Budde, Jennifer Neville, Pan Li, and Jianzhu Ma. 2021. Breaking the Limit of Graph Neural Networks by Improving the Assortativity of Graphs with Local Mixing Patterns. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467373

1 INTRODUCTION

A wide variety of complex systems spanning social, chemical, technological and biological domains are modelled using graphs (networks) where nodes represent concrete or abstract entities and edges symbolize pairwise node interactions. As more data are collected, many machine learning problems on graphs emerge, such as (semi-supervised) node classification, link prediction, graph classification and graph sampling [2, 6], which post us new computational challenges to develop graph-based machine learning models to address these problems. In recent years, graph neural networks (GNNs) [16, 51] with a message passing architecture [14] have gained significant attention from the research community due to their promising results on various graph related ML tasks.

In GNNs, the standard message passing works by propagating node features across edges and followed by aggregation viz. *sum, mean or attention* for a number of rounds [15, 20, 44]. The central idea is to utilize the neighbourhood information to construct a representation that can be beneficial for downstream learning tasks. Looking through the lens of graph signal processing, this operation of GNN could be viewed as a non-linear form of smoothing operation on the neighborhood or a low-pass graph filtering which is invariant to graph isomorphism. Clearly, one fundamental assumption made here is that similar nodes (w.r.t node attributes and labels) have a higher tendency to connect to each other compared to nodes that are far away. In other words, the philosophy followed is that proximity information from the surroundings of a node is a useful descriptor for predicting its labels. In network science, the concept *assortative mixing* is defined to quantify the degree of similar node attributes/labels aggregated on local network regions [29, 33]. For

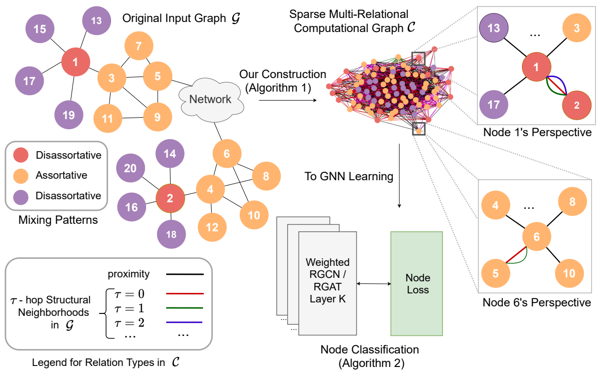
Original Input Graph $\mathcal{G}$
Sparse Multi-Relational Computational Graph $\mathcal{C}$
Our Construction (Algorithm 1)
Network
Node 1's Perspective
Node 6's Perspective
To GNN Learning
Disassortative
Assortative
Disassortative
Mixing Patterns
proximity
τ - hop Structural Neighborhoods in $\mathcal{G}$
$\tau = 0$
$\tau = 1$
$\tau = 2$
...
Legend for Relation Types in $\mathcal{C}$
Weighted RGCN / RGAT Layer K
Node Loss
Node Classification (Algorithm 2)

Figure 1: An input graph with diverse mixing pattern. Our pipeline uses both proximity and structural information to build a computation graph on which a GNN is run.

instance, in social networks, people with similar habits and ideals form friendships with each other [26]. In citation networks, papers from a similar area tend to cite each other. These can be recognized as *assortative mixing* nodes. In the opposite, *disassortative mixing* nodes can be found in technological networks where node hierarchy exists, heterosexual dating networks based on gender and ecological food webs where predator and prey tend to be dissimilar.

In this work, we aim to study the relationships between the limits of prediction performance of GNN and different mixing patterns in a graph. Quantifying mixing patterns in graphs has conventionally been done using a global binary notion of homophily/heterophily or assortative/disassortative mixing. These global summary statistics capture the average mixing patterns in the graph as an entire entity and are meaningful only when the mixing patterns of a whole network are centered around the mean. However, most real world graphs show heterogeneous and diverse mixing patterns wherein certain parts of the graph are assortative while others disassortative [5, 35]. For instance, Figure 2 demonstrates the distributions of assortativity on several real world graphs in which both multimodal distributions and long tail distributions are observed. Apparently, the global metrics will fail to measure this diversity on these complicated real world graphs. Recently, there has been a growing interest in designing new GNN models utilizing the nature of assortativity in graphs [8, 18, 27, 36, 52]. However, some of the representatives are based on heuristics leveraging node attributes [36] or intermediate node representations [27] to address disassortative nodes in the graph. Others simply incorporate node features from multi-hop neighbors [52] which might help improve predictions of disassortative nodes but at the same time suffer the over-smoothing problem of GNN [13, 25, 34]. GPR-GNN [8] may overcome the above issue using generalized graph diffusion [23] but loses model expressivity [7].

We reason that for GNNs to achieve good performance on graphs with diverse mixing, one has to provide sufficient inductive bias that lets the model adaptively choose either proximity, structural information or both for predicting node labels. This is based on the key observation that disassortative nodes (potentially far apart) may share similar structural features while assortative nodes tend to share similar features within their proximity. Consider the input graph $\mathcal{G}$ in Figure 1 and two nodes colored red i.e. 1 and 2 having same labels as each other but different from the labels of their own neighbourhoods. Based on the theory of mixing patterns, these two nodes are disassortative. Even though they are far apart, their local connecting pattern is quite similar. For instance, by comparing degree sequences of nodes 1 and 2 in $\mathcal{G}$ at various neighbourhoods, we can see that at 0-hop both have similar degree of 5 and 5 respectively, their 1-hop neighbours all have the same degree of 1 or 5 and so on. This shows nodes 1 and 2 are structurally quite similar and therefore, we can make use of their structural equivalences to construct a new graph in which nodes like 1 and 2 have a connection with large weight. On the other hand, consider node like 5 and 6. They mix assortatively and their surroundings/proximity can provide enough information to infer their labels. Further, we could still benefit from the fact that nodes 5 and 6 have similar local structure. In all, our idea is to construct a transformed computation graph that encodes both structure and proximity information w.r.t each node and the GNN is run on this computation graph instead of the original graph. Note that because similar (either structurally or proximity) nodes have large weight in the computation graph it has an enhanced level of assortativity and this can boost the prediction performance of GNNs.

To implement this idea, we first use a local measure of assortativity that can quantify diverse mixing patterns introduced in [35]. This new metric, named *local assortativity*, is a node-centric measure of mixing patterns that calculates assortativity within a local neighbourhood. We show that the representation capability of a wide range of GNN models is highly correlated with the level of local assortative mixing in the graph, which sets a limit to the prediction performance for GNN models based on message passing (Sec. 4). To break this limit, we then develop a new algorithm which can transform the input graph into a new one with higher assortativity level and suitable for the deployment of GNN by leveraging both proximity and the local structural similarity of nodes at multiple scales. (Sec. 5). Figure 1 shows the overall framework we propose based on the idea of transforming the input graph to a new computation graph on which the GNN is run. Lastly, we conduct extensive experiments and provide analysis to show the benefits of the proposed approach (Sec. 6). Our code and an easy to use tool for evaluating GNNs w.r.t network local assortativity is provided online [1].

2 PRELIMINARIES

A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which $\mathcal{V}$ and $\mathcal{E}$ denote the node set and edge set, respectively. An edge going from node $u \in \mathcal{V}$ to node $v \in \mathcal{V}$ is denoted as $(u, v) \in \mathcal{E}$. The adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is a convenient way to represent $\mathcal{G}$ where, $\boldsymbol{A}[u, v] = 1$ if $(u, v) \in \mathcal{E}$ otherwise 0. $\boldsymbol{A}$ is a real valued matrix when there are *weighted* edges. For *multi-relational* graphs, we extend the edge notation with type as $(u, v, \tau) \in \mathcal{E}$ to denote that the edge (u, v) belongs to type $\tau \in \mathcal{R}$. We represent the node *attributes* or *features* as a matrix $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$ where d is the feature dimension. The feature of a particular node $u \in \mathcal{V}$ is a vector $\boldsymbol{x}_u \in \mathbb{R}^d$. We denote the *neighbourhood* around a given node u which is a set of nodes

[1]https://github.com/susheels/gnns-and-local-assortativity

exactly one hop/step away as $\mathcal{N}(u) = \{v : (u, v) \in \mathcal{E}\}$. We consider the standard semi-supervised node classification task on $\mathcal{G}$, where each node $u \in \mathcal{V}$ has a class label y_u. The goal is to learn a function $f : \mathcal{V} \rightarrow \mathcal{Y}$ mapping the set of nodes to their class labels given some labelled nodes $\{(u_1, y_{u_1}), (u_2, y_{u_2}), \dots\}$ as training where $u_i \in \mathcal{V}$ and $y_{u_i} \in \mathcal{Y}$. **Neural Message Passing** is a framework that encompasses a range of GNN techniques inspired by the classical color refinement algorithm for graph isomorphism testing [14, 46]. During this refinement process, vector messages are passed between nodes across edges and updated using neural networks repeatedly for K rounds. The parameters are learned by defining a suitable loss function and followed by back propagation. A simplified and concrete working is given in the supplemental (Sec. A). The global *assortativity coefficient* r_{global} introduced by Newman [33] is used to measure **Mixing in Networks** which is the tendency of nodes with similar attributes/labels to be connected to other nodes. To characterize the mixing pattern, a quantity M_{gh} is defined to be the fraction of edges in a network that connect a node with label g to one of label h. This helps us define a *mixing matrix* $\boldsymbol{M}$ whose elements are M_{gh}. This matrix satisfies the following sum property $\sum_g \sum_h M_{gh} = 1$. Global assortativity is a summary statistic for the whole network and is defined as,

$$r_{\text{global}} = \frac{\sum_g M_{gg} - \sum_g a_g b_g}{1 - \sum_g a_g b_g} \tag{1}$$

where a_g and b_g represent the number of outgoing and incoming edges of all nodes of label g as follows, $a_g = \sum_h M_{gh}$ and $b_g = \sum_h M_{hg}$. The quantities a and b can be viewed as marginals that describe the proportion of edges starting from and ending at each of the attributes. For undirected graphs where ends of edges are of same type, quantities a_g and b_g are equal and $\boldsymbol{M}$ is symmetric. This allows us to write the elements of $\boldsymbol{M}$ as,

$$M_{gh} = \frac{1}{2m} \sum_{i:\tau_i = g} \sum_{j:\tau_j = h} A_{ij} \tag{2}$$

where A_{ij} is an element of adjacency matrix, $m = |\mathcal{E}|$ is the number of edges and τ_i represents the label of node i.

3 RELATED WORK

Graph Neural Networks (GNNs) have been successfully adopted for many graph related tasks [3, 12, 14, 45, 50] and much effort in the community has been in understanding the nature and working of GNNs either with the lens of signal processing [10, 13, 30, 34] or the combinatorial color refinement algorithm for graph isomorphism [24, 31, 40, 48]. Li et al. [25] points out that GNNs essentially enforce similarity of representations between adjacent nodes akin to some sort of *local smoothing*. In line with this view, NT and Maehara [34] shows that GNNs behave as "low-pass" filters filtering high frequency noise components in the convolution step. Fu et al. [13] theoretically characterize the behaviour of a number of GNN models by proposing that they work by smoothing and de-noising node features. All these results show that when node features and labels vary smoothly or in other words when there is assortative mixing, GNNs tend to work well.

Because the convolution operations are defined on neighbourhoods, the apparent local nature prohibits the use of higher-order information in the graph. To alleviate this, Li et al. [25] tried to stack multiple layers of GNNs but failed due to the over-smoothing problem resulting from node representations becoming indistinguishable. This problem has also been acknowledged in Klicpera et al. [21]. Another line of work proposes graph attention [18, 44] computed using node features however, they are still enforcing smoothing albeit adaptively making use of relevant information from a node's surrounding.

In light of these results, a few works propose to supplant the basic message passing framework of GNNs with extra graph information. PPNP [21] uses PageRank, GDC [22] utilizes graph diffusion (e.g.,heat kernels) and Geom-GCN [36] extends graph convolution with geometric aggregation derived by precomputing unsupervised node embeddings. GPR-GNN [8] allows different hop neighbors being associated with different signs of scalar weights to model high pass filters. Jumping Knowledge Networks [49] leverages different neighborhood ranges for each node to enable better structure-aware representations. Non-local GNNs [27] use attention to adaptively get relevant long range graph information while H_2GCN [52] and MixHop [1] directly include information from higher order neighbourhoods within each convolution step. A comprehensive review of various graph neural networks can be found in [6, 51].

4 GNNS AND LOCAL MIXING

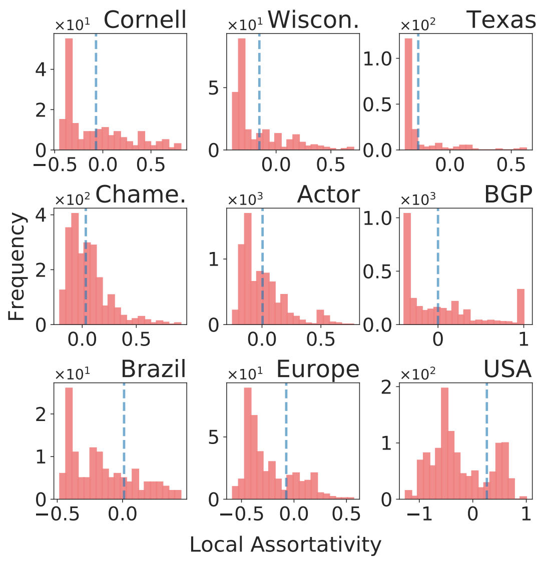

Figure 2: Observed distribution of node level assortativity in various graphs. The blue dotted line indicates global assortativity coefficient.

The global assortative coefficient r_{global} defined in Eq. 1 captures the average mixing pattern for the whole network but r_{global} is only meaningful if all nodes have mixing concentrated around the mean. It has been studied that real world graphs exhibit high variation in mixing patterns and we are essentially interested in how GNNs perform under such a diverse mixing. For this we first utilize a node level measure of assortativity r_{local} introduced by Peel et al. [35]

that is calculated w.r.t a local neighbourhood. This allows us to interpolate the mixing from individual nodes to global graph level by varying the size of the local neighbourhoods. Consider a simple random walker on an undirected graph. It walks by selecting an edge at i with an equal probability of A_{ij}/d_i where d_i is the degree of node i. Then, the stationary probability of being at node i is given by $\pi_i = d_i/2m$. This means each edge is traversed with a probability of $\pi_i A_{ij}/d_i = 1/2m$. Based on this and noting that $A_{ij} \in \{0, 1\}$, we can rewrite Eq. 2 as,

$$M_{gh} = \sum_{i:\tau_i=g} \sum_{j:\tau_j=h} \pi_i \frac{A_{ij}}{d_i} \tag{3}$$

Eq. 3 reinterprets the mixing $\boldsymbol{M}$ from the point of view of using random walk to visit the entire graph and thus reveals that the global assortativity counts all edges in the graph equally. For the local measure of assortativity the edges are weighted according to how local they are to a node of interest l by replacing the stationary distribution π_i with an alternative distribution over the nodes $w(i; l)$. Then Eq. 3 becomes,

$$M_{gh}(l) = \sum_{i:\tau_i=g} \sum_{j:\tau_j=h} w(i; l) \frac{A_{ij}}{d_i} \tag{4}$$

The personalized PageRank vector is utilized as a proxy for $w(i; l)$. Concretely, it is simple random walk with restarts i.e. during a simple random walk, the walker can return to the initial node of interest l with a probability of $(1 - \alpha)$. Varying α allows us to interpolate from the trivial local neighbourhood (when $\alpha = 0$ i.e. walker never leaves the node) to the global assortativity (when $\alpha = 1$ i.e. no restarts). Finally, the local assortativity metric for a node l parameterized by α is,

$$r_\alpha(l) = \frac{\sum_g M_{gg}(l) - \sum_g a_g^2}{1 - \sum_g a_g^2} \tag{5}$$

Note that we can recover the global assortativity metric from Eq. 5, $r_1(l) = r_{\text{global}}$ because when $(\alpha = 1)$ no restarts happen, $w_\alpha(i; l)$ falls back to π_i. When calculating the $r_\alpha(l)$ in practice, instead of choosing α heuristically, inspired by TotalRank [4], the PageRank vector is averaged over the entire range of $\alpha \in [0, 1]$ as,

$$w_{\text{tt}}(i; l) = \int_0^1 w_\alpha(i; l) d\alpha \tag{6}$$

With $w_{tt}(i; l)$ in place of $w(i; l)$ in Eq. 4 and finally using the resultant mixing matrix in Eq. 5 gives us our local assortativity metric $r_{\text{local}}(l)$.

In Figure 2, we examine various networks from different domains for existence of diverse mixing patterns using r_{local}. The details of these networks are given in the supplemental (Sec. C). In almost all of them we witness skewed and multimodal distributions. It is interesting to note that the global assortativity coefficient $r_{global} \approx 0$ (Eq. 1) while nodes exhibit diverse mixing over the full spectrum of r_{local}. This observation highlights that r_{global} is not necessarily reliable as it doesn't give a complete picture.

4.1 Analysis of GNNs Under Local Mixing

Given the observation in Figure 2, we are interested in studying the behaviour of various leading GNN models such as GCN [20], GIN [48] and GAT [44], when applied to graphs with diverse mixing patterns. The task of semi-supervised node classification is used as a proxy to understand the power of modelling graph data w.r.t different levels of local assortativity embedded in the graph. As shown in Figure 3, the performance of GNN models are highly correlated with the node-level local assortativity r_{local} within the same graph they are deployed. Across all the tested real world graphs, another clear pattern is that most of the popular GNN methods perform poorly for disassortative nodes l with $r_{\text{local}}(l) < 0$. The reason is that the features of disassortative nodes are vastly different from their neighbourhoods' and, GNN methods simply cannot create useful node representations based on the information provided by their neighbours through conventional node smoothing operation (alluded in theoretical works [13, 34]). We further characterize our reasoning below with the help of two definitions.

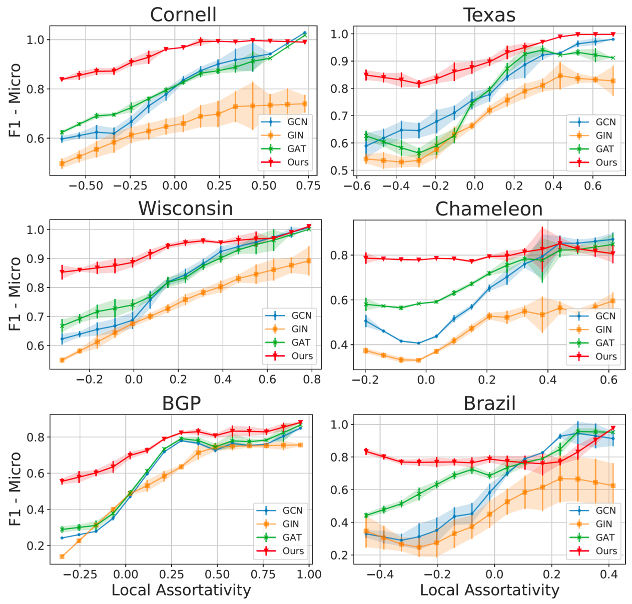

Figure 3: GNNs vs Local Assortativity on various graphs. Node classification is performed on various datasets (See Table 3 for details). Y-axis shows mean F1-Micro with standard deviation over 10 runs.

Definition 4.1 (Neighbourhood Label Smoothness). A node $u \in \mathcal{V}$ with class label $y_u \in \mathcal{Y}$ has label smoothness parameter defined on the neighbourhood $\mathcal{N}(u)$ as, $\epsilon_u = \frac{1}{|\mathcal{N}(u)|} \sum_{i \in \mathcal{N}(u)} P(y_i = y_u | y_u)$

Definition 4.2 (Neighbourhood Feature Smoothness). A node $u \in \mathcal{V}$ with feature vector $\boldsymbol{x}_u \in \mathbb{R}^d$ has a smoothness parameter defined on the neighbourhood $\mathcal{N}(u)$ as, $\lambda_u = ||\boldsymbol{x}_u - \frac{1}{|\mathcal{N}(u)|} \sum_{i \in \mathcal{N}(u)} \boldsymbol{x}_i||^2$

Labels of disassortative nodes and their neighbours have high probability of being different which means their labels tend to be not smooth in that neighbourhood, which is quite clear from Def. 4.1 (their ϵ_u's tends to be low). Then, it is also likely that features of such nodes are not smooth either (high λ_u). This is based on a reasonable assumption that features $\boldsymbol{x}_u$ and class labels y_u are correlated. However, GNNs try to smooth the node features in the latent space which in turn smooth out the class label predictions. Thus, performing poorly in a disassortative regime. On the flip side for neighbourhoods mixing assortatively, owing to smooth labels all across, ϵ_u will be large, which also means high feature smoothness (low λ_u), a regime that is very beneficial to GNNs. A key take away is that $\epsilon_u \propto 1/\lambda_u$ and this characterization explains the observations we witness in Figure 3.

5 OUR FRAMEWORK

In Section 4 we showed that real world graphs contain diverse mixing patterns and experimental results demonstrate that current GNN methods based on message passing are unable to provide good representations for nodes that show disassortative mixing. To break the limit introduced by the original graph structure, we design a graph transformation algorithm which can generate a new graph with higher assortativity. The central idea is to leverage the structural and proximity information in the original graph. Ideally, the resulted new graph should have the following properties: (1) Constructed from the original input graph using the same set of nodes without the class label information. (2) Encode the structural equivalences in the original input graph in a model free manner. (3) Encode proximity information as seen in the original graph.

These properties requires the construction of a new graph to be solely based on structural and proximity regularities of the original graph. Traditionally, GNN methods define convolutions on the original input graph and various design choices are made. Different from them, our framework aims to apply such GNN models on a transformed graph which we dub as the "computation graph" for the same machine learning task. Note that our framework consists of 2 stages. Stage (1) transforms the original input graph to a computation graph and in Stage (2) GNN methods are applied on the computation graph for learning. In the next section, we introduce one approach for graph transformation and later define message passing on the transformed graph.

5.1 Computation Graph

One obvious choice to encode structural equivalences between nodes is to compare ordered degree sequences at various hierarchies [37]. The rationale is that any two nodes with same degree are structurally similar, and if their one hop neighbours also have same degree, then they are even more structurally similar and so on. Therefore, a key observation is that the structural similarity between two nodes monotonically increases when their degree sequences get progressively similar. More formally, let $\mathcal{N}_\tau(g)$ denote the set of neighbouring nodes at exactly τ hops away from node g in graph $\mathcal{G}$. Let $s(V)$ represent a non-increasing (ordered) sequence of degrees of a set $V \subset \mathcal{V}$ of nodes. The goal is to compare ordered degree sequences at various neighbourhoods for every pair of nodes (g, h) in $\mathcal{G}$. The notion of *structural distance* [37] is recursively defined as follows,

$$f_\tau(g,h) = f_{\tau-1}(g,h) + \mathcal{D}(s(\mathcal{N}_\tau(g)), s(\mathcal{N}_\tau(h))) \quad (7)$$

where $\mathcal{D}(S_1, S_2) \geq 0$ measures the distance between ordered degree sequences S_1 and S_2 and $f_{-1}(\cdot) = 0$. $f_\tau(g,h)$ is defined for $\tau \geq 0$ and $|\mathcal{N}_\tau(g)|, |\mathcal{N}_\tau(h)| > 0$ i.e. only when neighbourhoods at τ exist. The cost function $\mathcal{D}(\cdot,\cdot)$ should ideally give small values for similar ordered degree sequences while provide large values for vastly different ones. Following [37], we make use of Fast Dynamic Time Warping (DTW) [39] that is best suited for loosely comparing sequences of different sizes. The recursive definition of structural distance in Eq. 7, makes sure that $f_\tau(\cdot,\cdot)$ can only increase as we successively progress through τ. So, for nodes g and h, that are structurally similar, structural distance is low.

Algorithm 1: Construction of computation graph

```
Input: Original Input Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
Hyper-Params.: No. of structural relations $T < dia(\mathcal{G})$
Output: Computation graph $C$
begin
    $\mathcal{E}' \leftarrow \emptyset$; $n \leftarrow |\mathcal{V}|$;
    $w_\tau \in \mathbb{R}^{n \times n} \leftarrow 0, \forall \tau \in \{0, 1 \cdots T\}$; $w_p \in \mathbb{R}^{n \times n} \leftarrow 0$;
    $f_{-1} \leftarrow 0$;
    for $(g, h) \in \mathcal{V}^2$ do
        for $\tau \in \{0, 1 \cdots T\}$ do
            $f_\tau(g,h) \leftarrow f_{\tau-1}(g,h) + \mathcal{D}(s(\mathcal{N}_\tau(g)), s(\mathcal{N}_\tau(h)))$;
            $w_\tau(g,h) \leftarrow e^{-f_\tau(g,h)}$
            $\mathcal{E}' \leftarrow \mathcal{E}' \cup (g, h, \tau)$;
        end
    end
    for $(g, h) \in \mathcal{E}$ do
        $\mathcal{E}' \leftarrow \mathcal{E}' \cup (g, h, p)$;  $w_p(g,h) \leftarrow 1$
    end
    $\mathcal{R} \leftarrow \{0, 1 \cdots T\} \cup p$;
    return $C = (\mathcal{V}, \mathcal{E}', \mathcal{R}, \{w_\tau, \forall \tau \in \mathcal{R}\})$
end
```

5.1.1 Graph construction. We construct a weighted multi-relational computation graph C which encodes the structural distances at various hierarchies between pairs of nodes and proximity information available in the original input graph $\mathcal{G}$. Let T denote a number less the diameter of the graph $\mathcal{G}$ and $n = |\mathcal{V}|$ the number of nodes. To construct the new graph C, we first add the original node set $\mathcal{V}$ and for each pair of nodes (g, h) we create $T + 1$ different types of edges where each type corresponds to the τ-hop neighbourhoods on which we calculated structural distance in $\mathcal{G}$. To encode structural equivalence between nodes we define edge weights $w_\tau(g,h)$ between g and h with structural relation type τ to vary inversely with structural distance $f_\tau(g,h)$ as follows:

$$w_\tau(g,h) = e^{-f_\tau(g,h)}, \quad \tau = 0, 1, \dots T \quad (8)$$

The edge weight for type τ between g and h is large when their τ-hop neighbors have similar network structure properties (low structural distance).

For proximity information, we simply use the original edges of $\mathcal{G}$ and add it to C with weight one. In total, this construction creates C which has $\mathcal{V}$ nodes and at most $\mathcal{E} + (T+1)\binom{n}{2}$ edges. The algorithm for the construction of this new graph is detailed in Algorithm 1. Naively using the above graph construction results in a large number of edges being introduced. For practicality, pairwise structural similarity calculations are restricted to $O(\log n)$ for each node at each τ-hop neighbourhood. We thus have $O(n \log n)$ edges for each τ instead of $O(n^2)$. For completeness we provide the efficient practical implementation in the supplement (Sec. D).

In this work, we provide one specific implementation of our general idea of using both structure and proximity information from the original graph into the computation graph. Other structural techniques viz. RolX [17], GraphWave [11] and generalized proximity inspired methods viz. Graph Diffusion [22], PageRank [21] can also be adopted.

5.2 Message Passing on the Multi-relational Computation Graph

The constructed multi-relational graph C and the original graph $\mathcal{G}$ share the same node set $\mathcal{V}$ and can be defined as $(\mathcal{V}, \mathcal{E}', \mathcal{R})$. Each edge in C going from node u to v of type τ is represented as a triplet $(u, v, \tau) \in \mathcal{E}'$. Recall that there are $T+1$ structural type edges and one proximity type edge. Thus, the total number of relations $|\mathcal{R}| = T + 1 + 1$. We now define the message passing procedure on C following the standard GNN formulation (Sec. A).

To account for the different relation types in C, following [41], we introduce a relation specific transformation matrix $\boldsymbol{W}_\tau$ for each type $\tau \in \mathcal{R}$ of edge and specify the AGGREGATE function as follows,

$$\boldsymbol{m}_u = \sum_{\tau \in \mathcal{R}} \sum_{v \in \mathcal{N}_1^\tau(u)} \boldsymbol{W}_\tau \, \boldsymbol{h}_v \; w_\tau(u,v) \; \alpha_\tau(u,v) \tag{9}$$

where $w_\tau(u,v)$ is the τ type specific edge weight between nodes u and v defined in Algo. 1. In line with our motivation of letting the model adaptively choose between structural and proximity information, we make use of an attention mechanism [44] defined using attention coefficients $e_\tau(u,v)$ that indicates the importance of node v's feature to node u w.r.t a particular relation type τ.

$$e_{u,v}^\tau = a_\tau(\boldsymbol{W}_\tau \boldsymbol{h}_u, \boldsymbol{W}_\tau \boldsymbol{h}_v) = \boldsymbol{a}^T [\boldsymbol{W}_\tau \boldsymbol{h}_u || \boldsymbol{W}_\tau \boldsymbol{h}_v] \tag{10}$$

where $a : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a shared attention mechanism parameterized by a learnable attention weight vector $\boldsymbol{a} \in \mathbb{R}^{2d}$, $\cdot^T$ is transpose operation and $||$ is concatenation operation. We inject the available computation graph information as follows,

$$\alpha_\tau(u,v) = \frac{\exp(\text{LeakyReLU}(e_{u,v}^\tau))}{\sum_{x \in \mathcal{N}_1^\tau(u)} \exp(\text{LeakyReLU}(e_{u,x}^\tau))} \tag{11}$$

Finally, The UPDATE function is defined as,

$$\boldsymbol{h}'_u = \sigma\Big(\boldsymbol{W}_{\text{self}} \, \boldsymbol{h}_u + \boldsymbol{W}_{\text{neig}} \, \boldsymbol{m}_u\Big) \tag{12}$$

We name our general model as WRGNN (weighted relational GNN) and from the above definitions, we select two model variants viz. WRGAT and WRGCN for experimental analysis which specifies if attention mechanism is used or not respectively. Algorithm 2 provides a procedure for applying such a K layer WRGNN on the computation graph for semi-supervised node classification.

6 EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of our framework against other methods under semi-supervised node classification setting. Note that our framework is quite flexible so any GNN model based on message passing could be adopted on our computation graph. To evaluate the performance of our method, we use real world graphs from different domains viz. Hyperlinked Web Pages [36], Citation Networks [32], Air Traffic Networks [37] and Internet's Inter-Domain Routing Network [18, 28]. These graphs are know to exhibit diverse mixing as we showed in Sec. 4 and thus provides us with the means to assess GNN based methods. Statistics and brief descriptions are provided in Table 3 and Sec. C.

6.1 Baseline Methods and Experiment Setup

We primarily consider methods that utilize GNN models which adopted messaging passing operation as their main backbones. GCN [20] and GraphSage [15] are methods where convolutions are strictly based on first order neighbour aggregation scheme for each layer. GCN-Cheby [9] generalizes convolutions with the help of k-hop localized spectral filters. GAT [44] adaptively aggregates immediate neighbour information using attention coefficients which are also derived from node features. MixHop [1] and H_2GCN [52] generalize the node aggregation beyond the first order neighbourhoods and dynamically considers node features k-hops away. It is important to note that all baseline methods operate on the original graph and thus have access to only proximity information albeit in different forms.

We perform semi-supervised node classification and use the classification accuracy and F1-Micro scores as performance metrics to evaluate different approaches. The training/validation/testing data splits for all the methods to be compared is shown in Table 3. For Hyperlinked Web Page and Citation Networks, we report the performance of mean ± std. dev. on 10 random splits provided by Pei et al. [36] which is available on their GitHub [2]. Reported values for Air Traffic Networks and BGP Networks are based on 20 and 10 random splits respectively. All the implemented methods including our own are all trained until the loss function converges and the final models are selected based on the prediction performance on the validation sets. The sensitivity analysis and hyper-parameter search is performed on the validation set and more details are provided in the supplement (Sec. F).

6.2 Local Assortativity Distribution Shift

We first perform a quick study to confirm our graph transformation algorithm indeed enhances the level of local assortativity in comparison to the original input graph. To achieve this, we focus on all the disassortative nodes ($r_{\text{local}} < 0$) from the input graph $\mathcal{G}$ and track how they mix in the transformed computation graph C. From Figure 4, a clear distribution shift of local assortativity could be observed. That is, the previously disassortative nodes in G, are more assortative in the new computation graph after transformation. This empirically verifies the claim we raised earlier about similar structural regularities between a pair of nodes being captured in the computation graph as a result of our transformation (hence the increased assortativity). In addition, we have a reason to believe that performance improvement of our relational GNN model variants WRGCN and WRGAT are rooted in the increase of local assortativity for disassortative nodes in the original graph. This is clearly witnessed in Figure 3 (shown in red).

[2] https://github.com/graphdml-uiuc-jlu/geom-gcn/tree/master/splits

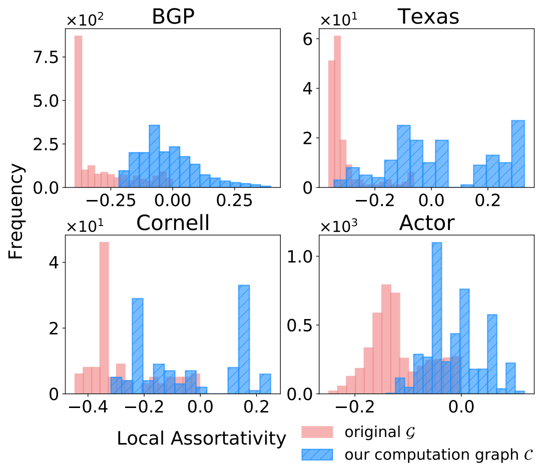

Figure 4: Distributions of node level assortativity for the original graph and the newly constructed computation graph on different datasets.

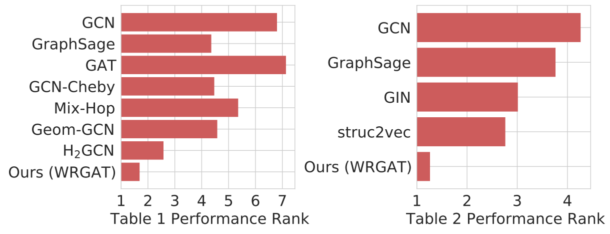

Figure 5: Overall perf. rank of various methods for datasets in Table 1 & 2. Lower rank signifies better performance

6.3 Performance of Node Classification

Hyperlinked Web Page and Citation Networks. Table 1 shows the evaluation of our framework against other GNN based baselines on the node classification task for these datasets. Mean test accuracy and standard deviation numbers are reported for each method. Values for H_2GCN [52]and Geom-GCN [36] are taken from the respective papers. We use the same train/val/test splits as [36, 52] for comparability. We find that our framework consistently performs well for the Hyperlinked Web Page Networks owing to the rich structural regularities that our computation graph captures. On Citation Networks which predominantly have assortative mixing, our framework gives comparable performance to baselines. H_2GCN and MixHop utilize higher order neighbourhoods in each convolution which does help over standard GNN methods, but can also be a lot harder to train and they also suffer from oversmoothing problem. Our framework makes it possible to define computation graphs that can directly tap into the structural regularities thereby making effective use of graph information, while the message passing defined on such a computation graph takes care of adapting to node features. Overall performance rank of various methods in Table 1 on both disassortative and assortative datasets is shown in Figure 5. It is clear that our model variant WRGAT run on the computation graph achieves the lowest rank overall hence supporting

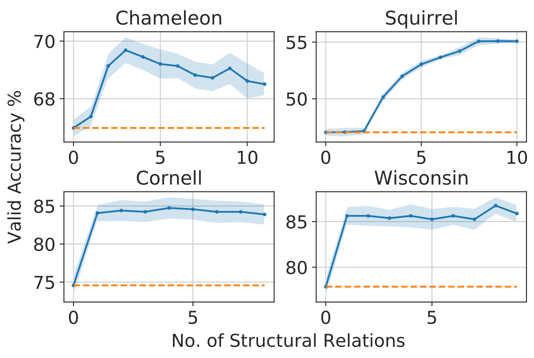

Figure 6: Sensitivity w.r.t structure relations. Baseline is using only proximity information.

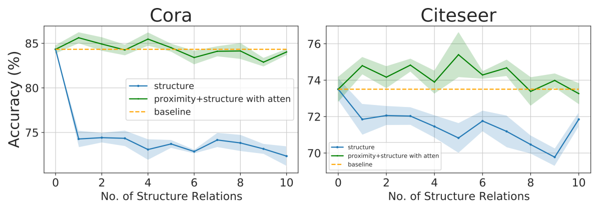

Figure 7: Sensitivity w.r.t structure relations. Baseline is using only proximity information.

our claim of achieving superior performance in both disassortative and assortative regime.

Air Traffic and BGP Networks. Table 2 gives comparisons of our framework against other GNN methods for node classification on multiple Air Traffic Networks (ATNs) and the Internet Domain Network (BGP). It is clear from the table that, our framework achieves strong performance compared to other baselines. ATNs don't have node attributes and baseline GNN methods perform poorly while our framework utilizing structure is better suited for the task. Strong performance is seen because our construction of the computation graph explicitly looks at different neighbourhoods that is very beneficial in air traffic networks. Major hubs connect to local airports (disassortative) while they also mix with other hubs (assortative). Structure alone is capable of capturing this diversity and our computation graph takes a step in that direction. The BGP network also exhibits diverse mixing and we believe that the strong performance is due to the adaptive selection of both proximity and structural information.

6.4 Sensitivity Analysis

We provide sensitivity analysis based on validation split w.r.t the number of structural relations used in our computation graph, which is our main hyper-parameter. Figure 6 shows the analysis for disassortative networks and is quite clear that adding structural information at various scales significantly improves validation performance when compared to baseline of just using proximity information. Figure 7 shows an interesting picture for highly assortative networks. Here, the take away is that structure alone is not useful

Table 1: Semi-supervised node classification showing mean test accuracy ± std. over 10 runs. Club Suit [♣] denotes result obtained from the best model variant of respective papers.

	Chameleon	Squirrel	Actor	Cornell	Texas	Wisconsin	Cora	Citeseer	Pubmed
GCN	59.82 ± 2.58	36.89±1.34	30.26±0.79	57.03±4.67	59.46±5.25	59.80±6.99	87.28±1.26	76.68±1.64	87.38±0.66
GraphSage	58.73±1.68	41.61±0.74	34.23±0.99	75.95±5.01	82.43±6.14	81.18±5.56	86.90±1.04	76.04±1.30	88.45±0.50
GAT	54.69±1.95	30.62±2.11	26.28±1.73	58.92±3.32	58.38±4.45	55.29±8.71	86.37±1.69	75.46±1.72	87.62±0.42
GCN-Cheby	55.24±2.76	43.86±1.64	34.11±1.09	74.32±7.46	77.30±4.07	79.41±4.46	86.86±0.96	76.25±1.76	88.08±0.52
MixHop	60.50±2.53	43.80±1.48	32.22±2.34	73.51±6.34	77.84±7.73	75.88±4.90	83.10±2.03	70.75±2.95	80.75±2.29
Geom-GCN ♣	60.90	38.14	31.63	60.81	67.57	64.12	85.27	77.99	90.05
H_2GCN ♣	59.39±1.98	37.90±2.02	35.86±1.03	82.16±4.80	84.86±6.77	86.67±4.69	87.67±1.42	76.72±1.50	88.50±0.64
Ours (WRGAT)	65.24±0.87	48.85±0.78	36.53±0.77	81.62±3.90	83.62±5.50	86.98±3.78	88.20±2.26	76.81±1.89	88.52±0.92

Table 2: Node classification on Air Traffic Networks and BGP Network. Mean test acccuracy ± std. is shown over 20 runs.

	Brazil	Europe	USA	BGP
GCN	64.55±4.18	54.83±2.69	56.58±1.11	53.33±0.18
GraphSage	70.65±5.33	56.29±3.21	50.85±2.83	65.19±0.28
GIN	71.89±3.60	57.05±4.08	58.87±2.12	49.51±1.52
Struc2vec	70.88±4.26	57.94±4.01	61.92±2.61	48.40±1.39
Ours	76.92±5.45	57.12±2.81	63.02±1.87	66.54±0.48

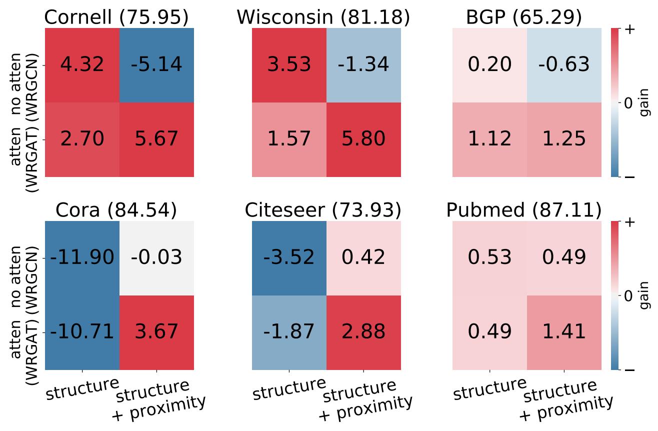

Figure 8: Ablation w.r.t different model and graph information choices. Baseline (proximity only) is shown in parenthesis. Numbers indicate gain over baseline.

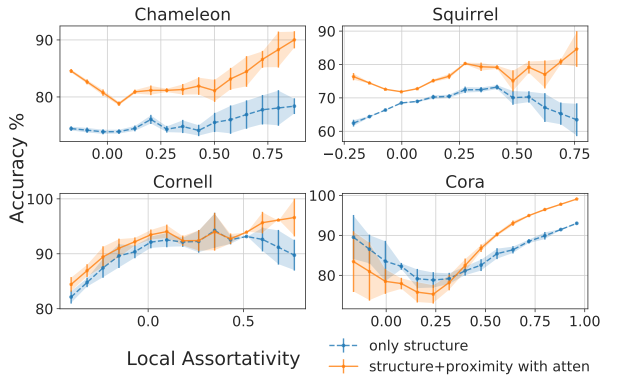

Figure 9: Ablation w.r.t graph info and local assortativity

(blue line), while an adaptive structure+proximity with attention model i.e. using WRGAT (green line) is able to give better consistent performance over proximity only baseline.

6.5 Ablation Analysis

Figure 8 provides ablation analysis for a number of networks and shows test performance gains over proximity only baseline for our model variants (WRGCN and WRGAT) in y-axis and against the use of either proximity, structure or both kinds of information in x-axis. For the top row consisting of predominantly disassortative networks, the take away is that structure only information is capable of providing better performance over proximity only information, but when both kinds of information is available, the attention model is best suited. For the bottom row consisting of highly assortative networks, clearly structure only information hurts performance compared to baseline. We reason that structure becomes less important for graphs with high assortativity however, when both kinds of information is available and attention is used we witness the best gains. These results further supports our model choices.

The previous ablation study shows test performance gains for the whole dataset. We now study how these different model configurations behave w.r.t node level local assortativity we introduced earlier. With Figure 9 we are able to study their behaviour in finer detail due to the notion of local assortativity. It supports our hypothesis that in the high local assortativity regime, proximity information dominates in discriminatory power and structure only information leads to worsened performance as it becomes irrelevant. However when using both kinds with an adaptive mechanism we can see increased performance over the full spectrum.

7 CONCLUSION

The level of mixing plays a crucial role in characterizing real world networks. In this work we have used the quantification of local mixing patterns to study the predictive performance of graph neural networks. Our observations and results offer a new perspective to study the limitations of GNNs. Motivated by the findings of our analysis, we develop a graph transformation technique that has shown to experimentally improve assortativity owing to the use of structural regularities in the input graph and there by increasing GNN performance. Extensive experiments on various real world networks from different domains supports our claim of running GNNs on a transformed computation graph and adaptively choosing between structure and proximity information rather than on the original. The connections we find with mixing patterns and GNN learnability provides motivation for future work to provide possible theoretical claims relating the two. We also hope that this study leads to the creation of other benchmark datasets with diverse mixing patterns which can aid in the robust evaluation of future GNN methods.

ACKNOWLEDGMENTS

This research is supported by the National Science Foundation under contract numbers CCF-1918483, IIS-1618690, and CCF-0939370.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. *ICML* (2019).

[2] Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. 2013. Network sampling: From static to streaming graphs. *TKDD* (2013).

[3] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2018. Learning to represent programs with graphs. *ICLR* (2018).

[4] Paolo Boldi. 2005. TotalRank: Ranking without damping. In *Special interest tracks and posters of the 14th international conference on World Wide Web.*

[5] George T Cantwell and MEJ Newman. 2019. Mixing patterns and individual differences in networks. *Physical Review E* (2019).

[6] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. 2020. Machine Learning on Graphs: A Model and Comprehensive Taxonomy. *arXiv preprint arXiv:2005.03675* (2020).

[7] Lei Chen, Zhengdao Chen, and Joan Bruna. 2020. On Graph Neural Networks versus Graph-Augmented MLPs. *arXiv preprint arXiv:2010.15116* (2020).

[8] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations. https://openreview. net/forum.*

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS.*

[10] Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard. 2020. Graph signal processing for machine learning: A review and new perspectives. *arXiv preprint arXiv:2007.16061* (2020).

[11] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning structural node embeddings via diffusion wavelets. In *KDD.*

[12] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS.*

[13] Guoji Fu, Yifan Hou, Jian Zhang, Kaili Ma, Barakeel Fanseu Kamhoua, and James Cheng. 2020. Understanding Graph Neural Networks from Graph Signal Denoising Perspectives. *arXiv preprint arXiv:2006.04386* (2020).

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML.*

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS.*

[16] William L Hamilton. 2020. Graph representation learning. (2020).

[17] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. Rolx: structural role extraction & mining in large graphs. In *KDD.*

[18] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. 2019. Measuring and improving the use of graph information in graph neural networks. In *ICLR.*

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. *ICLR* (2019).

[22] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. In *NeurIPS.*

[23] Pan Li, Eli Chien, and Olgica Milenkovic. 2019. Optimizing generalized pagerank methods for seed-expansion community detection. *Advances in Neural Information Processing Systems* 32 (2019).

[24] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance Encoding–Design Provably More Powerful GNNs for Structural Representation Learning. *NeurIPS* (2020).

[25] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *arXiv preprint arXiv:1801.07606* (2018).

[26] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* (2007).

[27] Meng Liu, Zhengyang Wang, and Shuiwang Ji. 2020. Non-Local Graph Neural Networks. *arXiv preprint arXiv:2005.14612* (2020).

[28] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and KC Claffy. 2013. AS relationships, customer cones, and validation. In *Proceedings of the 2013 conference on Internet measurement conference.*

[29] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* (2001).

[30] Yimeng Min, Frederik Wenkel, and Guy Wolf. 2020. Scattering GCN: Overcoming Oversmoothness in Graph Convolutional Networks. *arXiv preprint arXiv:2003.08414* (2020).

[31] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI.*

[32] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. 2012. Query-driven active surveying for collective classification.

[33] Mark EJ Newman. 2003. Mixing patterns in networks. *Physical review E* 67, 2 (2003), 026126.

[34] Hoang NT and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).

[35] Leto Peel, Jean-Charles Delvenne, and Renaud Lambiotte. 2018. Multiscale mixing patterns in networks. *PNAS* (2018).

[36] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *ICLR* (2020).

[37] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *KDD.*

[38] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale attributed node embedding. *arXiv preprint arXiv:1909.13021* (2019).

[39] Stan Salvador and Philip Chan. 2007. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intell. Data Anal.* (2007).

[40] Ryoma Sato. 2020. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078* (2020).

[41] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference.*

[42] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* (2008).

[43] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social Influence Analysis in Large-Scale Networks. In *KDD.*

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

[45] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions On Graphics (tog)* (2019).

[46] B. Y. Weisfeiler and A. Leman. 1968. Reduction of a graph to a canonical form and an algebra arising during this reduction (in Russian). *Nauchno-Technicheskaya Informatsia* (1968).

[47] Jun Wu, Jingrui He, and Jiejun Xu. 2019. Net: Degree-specific graph neural networks for node and graph classification. In *KDD.*

[48] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *ICLR* (2018).

[49] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. *ICML* (2018).

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

[51] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep learning on graphs: A survey. *IEEE TKDE* (2020).

[52] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. *NeurIPS* (2020).

A BACKGROUND

Neural Message Passing Concretely, during the k^{th} iteration a *hidden representation* $\boldsymbol{h}_u^{(k-1)}$ corresponding to each node $u \in \mathcal{V}$ is updated using u's neighbourhood information. Expressed as,

$$\boldsymbol{m}_{\mathcal{N}(u)}^k = \text{AGGREGATE}^k\left(\{\boldsymbol{h}_v^{(k-1)} : v \in \mathcal{N}(u)\}\right) \quad (13)$$

$$\boldsymbol{h}_u^{(k)} = \text{UPDATE}^k\left(\boldsymbol{h}_u^{(k-1)},\ \boldsymbol{m}_{\mathcal{N}(u)}^k\right) \quad (14)$$

where AGGREGATE$(\cdot)$ is a trainable differentiable function mapping sets of hidden node representations of u's neighbours to an aggregated message vector, UPDATE$(\cdot)$ is also a trainable differentiable function that maps both u's current hidden representation and the aggregated message vector to u's updated representation. The initial representation $\boldsymbol{h}_u^{(0)}$ is initialized using the original node feature $\boldsymbol{x}_u$. After a total of K iterations following Eq. 13 and 14, we obtain final representations $\boldsymbol{z}_u = \boldsymbol{h}_u^K, \forall u \in \mathcal{V}$. To perform node classification, we derive the class label of a node u by decoding it's final representation via, $y_u = \text{argmax}(\text{softmax}(\text{MLP}_\theta(\boldsymbol{z}_u)))$ where MLP is a neural network with trainable parameters θ and the softmax function is used to get a probability distribution over the classes.

B APPLYING GNN ON COMPUTATION GRAPH

Algorithm 2: Procedure for Node Classification

```
Input: Computation Graph $C = (\mathcal{V}, \mathcal{E}', \mathcal{R}, \{w_\tau, \forall\tau \in \mathcal{R}\})$;
       Node Features $\{\boldsymbol{x}_u \in \mathbb{R}^d, \forall u \in \mathcal{V}\}$;
       1-hop $\tau$-relation specific Neighbourhood Function
       on $C$ as $\mathcal{N}_1^\tau(u) = \{v : (u, v, \tau) \in \mathcal{E}'\}$;
Output: Predicted node labels
1  begin
2      $\boldsymbol{h}_u^0 \leftarrow \boldsymbol{x}_u, \forall u \in \mathcal{V}$;
3      for $k = 1, \dots K$ do
4          for $u \in \mathcal{V}$ do
5              $\boldsymbol{m}_u^k \leftarrow \sum_{\tau \in \mathcal{R}} \sum_{v \in \mathcal{N}_1^\tau(u)} \boldsymbol{W}_\tau^k\, \boldsymbol{h}_v^{k-1}\, w_\tau(u,v)\, \alpha_\tau(u,v)$;
6              $\boldsymbol{h}_u^k \leftarrow \sigma\left(\boldsymbol{W}_{\text{self}}^k\, \boldsymbol{h}_u^{k-1} + \boldsymbol{W}_{\text{neig}}^k\, \boldsymbol{m}_u^k\right)$
7          end
8          $\boldsymbol{h}_u^k \leftarrow \boldsymbol{h}_u^k / ||\boldsymbol{h}_u^k||_2,\ \forall u \in \mathcal{V}$;
9      end
10     $z_u \leftarrow \boldsymbol{h}_u^K,\ \forall u \in \mathcal{V}$;
       // Predict node labels
11     for $u \in \mathcal{V}$ do
12         $\boldsymbol{p}_u \leftarrow \text{softmax}(\text{MLP}_\theta(\boldsymbol{z}_u))$;
13         $y_u \leftarrow \text{argmax}(\boldsymbol{p}_u)$
14     end
15     return $y_u, \forall u \in \mathcal{V}$
16 end
```

C DATASET

Table 3 provides the statistics of the datasets we use for evaluation. We select a wide range of frequently evaluated datasets from different domains and below we provide brief descriptions.

Chameleon and Squirrel collected by [38] are networks of hyperlinked web pages on Wikipedia related to animal topics. The nodes (here pages) are labelled from one of 5 classes based on the average traffic (views) they received. Node features are bag-of-words representation of nouns in the respective pages. We download the processed data from Pei et al. [36].

Actor is a co-occurrence network based on the film-director-actor-writer network from [43]. In this dataset, nodes represent actor web pages on Wikipedia and edges symbolize co-occurrence on the same web page. Node features are bag-of-words representation of the corresponding pages and labels are placed according to topics on actor web page. The dataset is from Pei et al. [36].

Cornell, Texas and Wisconsin collected as part of CMU WebKB project. Nodes are university web pages and edges are hyperlinks between them. Node labels are one of student, project, course, staff or faculty. Node features are bag-of-words representation of the corresponding web pages. The dataset is also from Pei et al. [36].

Cora, Citeseer and Pubmed introduced by [32, 42] are citation networks where node represent scientific papers and edges are citation relationships. Node features are bag-of-words representation of the paper and labels are the scientific field they represent.

Air Traffic Networks from three regions **Brazil, Europe and USA** is collected by the respective civil aviation agencies. Nodes represent airports and edges mean the presence of commercial routes between nodes. Nodes are labelled according to the traffic (aircraft landings and takeoffs) or level of activity (by passenger count) an airport witnesses. We get the data from Ribeiro et al. [37].

BGP Network collected by [28] represents the inter-domain structure of the Internet. Nodes represent autonomous systems and edges indicate business relationships between nodes. Node features contain categorical location and topology information and labels are based on the type or tier of the autonomous system. We get the processed data from Hou et al. [18].

D PRACTICAL COMPUTATION GRAPH CONSTRUCTION

Algo. 1 when naively run requires $O(n^2)$ structural similarity calculations. However in practice we use an heuristic algorithm which achieves $O(n \log n)$ calculations. The intuition is that we don't need to look at node pairs with large degree differences. For instance, given nodes u and v with degree 1 and 20, we don't have to compute the similarity between u and v as their structural similarity will be extremely small. We cap ourselves to a budget of $O(\log n)$ nodes to look at for each node. The budget is picked based on the heuristic of most similar degrees. The complete practical procedure is shown in Algorithm 3.

E COMPARISON AGAINST STRUCTURE AWARE GNNS

We consider structure aware degree-based GNN models i.e. DEMO-Net [47] as a baseline for node classification accuracy. DEMO-Net uses an equal number of train/val/test nodes i.e. 33% each for the

Table 3: Dataset statistics. Details of ★ discussed in Sec. 6.1.

	Hyperlinked Web Pages Network						Citation Network			Air Traffic Network			Internet Network
	Chameleon	Squirrel	Actor	Cornell	Texas	Wisconsin	Cora	Citeseer	Pubmed	Brazil	Europe	USA	BGP (small)
#Nodes $\|\mathcal{V}\|$	2,277	5,201	7,600	183	183	251	2,708	3,327	19,717	131	399	1,190	10,176
#Edges $\|\mathcal{E}\|$	31,421	198,493	26,752	280	295	466	5,429	4,732	44,338	1,038	5,995	13,599	206,799
#Classes $\|\mathcal{Y}\|$	5	5	5	5	5	5	7	6	3	4	4	4	7
#Node Features d	2,325	2,089	931	1,703	1,703	1,703	1,433	3,703	500	1	1	1	287
Assortativity r_{global}	0.0331	0.0070	0.0047	-0.0706	-0.2587	-0.1524	0.7710	0.6713	0.6860	0.0116	-0.0737	0.2629	0.0029
Train/Val/Test Splits ★	60/20/20						60/20/20			80/10/10			70/10/20

Algorithm 3: Efficient construction of computation graph

```
Input: Original Input Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$;
       $\tau$-hop Neighborhood Function on $\mathcal{G}$
       as $\mathcal{N}_\tau(u) : u \to 2^{\mathcal{V}}$; Ordered Degree Sequence
       Function $s(V), V \subset \mathcal{V}$; Sequence Comparison Cost
       Function $\mathcal{D}$
Hyper-Params.: No. of structural relations $T < dia(\mathcal{G})$
Output: Computation graph $C$
1  begin
2      $\mathcal{E}' \leftarrow \emptyset$; $n \leftarrow |\mathcal{V}|$;
3      $w_\tau \in \mathbb{R}^{n\times n} \leftarrow 0, \forall \tau \in \{0, 1 \cdots T\}$; $w_p \in \mathbb{R}^{n\times n} \leftarrow 0$;
4      $f_{-1} \leftarrow 0$;
5      $S \leftarrow [degree(u)]\ \forall u \in \mathcal{V}$ ;
6      Sort S;                                   // $O(n \log n)$
7      for $g \in \mathcal{V}$ do
8          $pos \leftarrow$ BinarySearch($S, degree(g)$);      // $O(\log n)$
9          $P \leftarrow \log(n)$ positions left and right of $pos$ in $S$;
10         for $h \in P$ do
               // P contains $O(\log n)$ nodes
11             for $\tau \in \{0, 1 \cdots T\}$ do
                   // Calculate structural dist. Eq. 7
12                 $f_\tau(g, h) \leftarrow$
                     $f_{\tau-1}(g, h) + \mathcal{D}(s(\mathcal{N}_\tau(g)), s(\mathcal{N}_\tau(h)))$;
                   // Calculate edge weights.
13                 $w_\tau(g, h) \leftarrow e^{-f_\tau(g,h)}$ ;           // Eq. 8
                   // Extend the edge set
14                 $\mathcal{E}' \leftarrow \mathcal{E}' \cup (g, h, \tau)$;
15             end
16         end
17     end
18     for $(g, h) \in \mathcal{E}$ do
19         $\mathcal{E}' \leftarrow \mathcal{E}' \cup (g, h, p)$;  $w_p(g, h) \leftarrow 1$
20     end
21     $\mathcal{R} \leftarrow \{0, 1 \cdots T\} \cup p$;
22     return $C = (\mathcal{V}, \mathcal{E}', \mathcal{R}, \{w_\tau, \forall \tau \in \mathcal{R}\})$
23 end
```

original results (Table 5 of their paper) instead of our (80/10/10) % split (as also done in struc2vec). For fair comparison we rerun our model with the exact split and setup as demo-net. The results in Table 4 show that our graph transformation followed by GNN approach is superior compared to structure aware GNN baseline.

Table 4: Comparison against DEMO-Net with 33 % each train/val/test split. Node classification accuracy mean ± standard variance (same metric as DEMO-Net [47])

	Brazil	Europe	USA
DEMO-Net (hash)	0.614± 0.069	0.479±0.064	0.659±0.020
DEMO-Net (weight)	0.543±0.034	0.459±0.025	0.647±0.021
Ours	0.655±0.0055	0.539±0.0013	0.636±0.0043

Although DEMO-Net utilizes degree information in message passing, the architecture is quite similar to a GIN [48] model. In terms of expressive power, degree comparisons are usually limited to 2-3 hops as going higher leads to over-smoothing problems. Our graph transformation explicitly compares degrees at arbitrary hops and thereby provides a more efficient means to improve the assortativity of the graph.

F HYPER-PARAMETER SETTING

For all our experiments, we use Adam [19] algorithm with learning rate of $\{1e-2, 1e-3\}$ and weight decay of $\{0, 1e-5, 5e-4, 5e-6\}$ to optimize our model. Our WRGNN model variants contains 2 layers with an another fully connected MLP$_\theta$ on top of it. We select ReLU as the nonlinear activation. For the model with attention mechanism i.e. WRGAT, we use LeakyReLU with a negative input slope of 0.2. We sweep all the hidden dimensions from $\{16, 32, 64, 128\}$ for the WRGNN layer and $\{32, 64, 128\}$ for the final MLP$_\theta$ layer using cross validation. We set the maximum learning epochs as 500 with early stopping parameter 100. Specifically, for Hyperlinked Web Page Networks (Table 1) dropout operation with a probability of 0.8 is applied on each WRGNN layer. For BGP Network our model uses dropout of 0.5, learning rate as 1e-2 with weight decay of 0. Finally, for Air Traffic Networks (Table 2), dropout is set to 0.6, learning rate is 1e-3 with weight decay 5e-6 and T is set to 5, 5 and 8 for Brazil, Europe and USA datasets, respectively. The hyper-parameter T related to structural similarity calculations is chosen based on the validation set. Its sensitivity against validation accuracy for some datasets are provided in Figures 6 and 7.