Eigen-GNN: a Graph Structure Preserving Plug-in for GNNs

Ziwei Zhang, Peng Cui, Jian Pei ,*Fellow, IEEE,* Xin Wang, and Wenwu Zhu, *Fellow, IEEE*

Abstract—Graph Neural Networks (GNNs) are emerging machine learning models on graphs. Although sufficiently deep GNNs are shown theoretically capable of fully preserving graph structures, most existing GNN models in practice are shallow and essentially feature-centric. We show empirically and analytically that the existing shallow GNNs cannot preserve graph structures well. To overcome this fundamental challenge, we propose Eigen-GNN, a simple yet effective and general plug-in module to boost GNNs ability in preserving graph structures. Specifically, we integrate the eigenspace of graph structures with GNNs by treating GNNs as a type of dimensionality reduction and expanding the initial dimensionality reduction bases. Without needing to increase depths, Eigen-GNN possesses more flexibilities in handling both feature-driven and structure-driven tasks since the initial bases contain both node features and graph structures. We present extensive experimental results to demonstrate the effectiveness of Eigen-GNN for tasks including node classification, link prediction, and graph isomorphism tests.

Index Terms—Graph Neural Networks, Eigenvector, Graph Structure, Dimensionality Reduction

1 Introduction

Graphs are natural representations for complex data that cannot be represented well using simpler data structures, such as social networks, biomedical graphs, and traffic networks. In a graph, the nodes represent objects, and the edges represent relations between objects. Besides carrying relation information through graph structures, graphs are often associated with rich content information such as attributes of nodes. Content (features) and structures often provide information complementary to each other. In different analytics tasks, content and structures play different roles. Some analytics tasks focus on the content information, e.g., in document topic classifications, the content of documents usually provides dominant information. We call such tasks *feature-driven*. In some other analytics tasks, structures are the major player. A great example of such *structure-driven tasks* is influence analysis in social networks. Of course, there are always some analytics tasks where both content and structure information are needed. For example, in social recommendations, both user profiles (content) and user interactions (structure) are indispensable in understanding user preferences.

Recently, Graph Neural Networks (GNNs) are emerging machine learning models on graphs and are expected to provide a unified framework to deal with features and structures simultaneously. For example, in the message-passing framework [1], nodes exchange information with their neighbors in each step to update their feature information. In this way, GNNs model node attributes and graph structures in an end-to-end learning architecture.

It has been proven theoretically that GNNs with a sufficiently large number of layers can fully preserve many important graph structures such as the limiting distribution of a random walk on graphs [2], [3], graph moments of any order [4], or even universal approximations under certain conditions [5], [6]. However, training deep GNNs suffers from many practical challenges, such as over-smoothing [7], [8]. In practice, most successful GNNs are shallow, having no more than three or four layers [9].

However, shallow GNNs are distant from those theoretically expressive GNNs that have a large number of layers. Theoretical analysis shows that the existing shallow GNNs essentially are feature-centric, i.e., node attributes play major roles, and graph structures only provide auxiliary information. For example, Li *et al.* [7] analyzed GNNs as a special form of Laplacian smoothing of node attributes. Maehara [10] and Wu *et al.* [11] showed that GNNs are equivalent to a low-pass filter by treating node features as graph signals. Given these discussions showing the strength of GNNs in preserving features, a critical question is *whether the shallow GNNs in practice can sufficiently preserve graph structures*, which motivates this study.

To answer this question, we first report experimental analysis on a series of synthetic datasets (please refer to Section 3.1 for details). We observe consistent results with the analysis mentioned above: in the structure-driven tasks where graph structures are heavily involved, the existing shallow GNNs have poor performance. We further examine this observation by treating GNNs as a type of dimensionality reduction process. We show that the features of nodes provide the initial bases for the dimensionality reduction, making the resulted predictions of the existing shallow GNNs tend to be feature-centric. Therefore, the existing shallow GNNs are incapable of sufficiently preserving graph structures in practice.

Can we have a simple and general mechanism to empower the practical shallow GNNs to preserve graph structures well? To

- *Z. Zhang, P. Cui, X. Wang, and W. Zhu are with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. E-mail: {zwzhang, cuip, xin_wang, wwzhu} @tsinghua.edu.cn*
- *J. Pei is with the School of Computing Science at Simon Fraser University, Burnaby, BC V5A 1S6, Canada. E-mail: jpei@cs.sfu.ca*

tackle this fundamental challenge, we propose Eigen-GNN[1], a simple yet effective and general plug-in module to boost GNNs ability in preserving graph structures. Specifically, we integrate the eigenspace of graph structures with GNNs by concatenating the eigenvectors of a graph structure matrix to the node attributes. In this way, since the initial bases are expanded to contain both node features and graph structures, Eigen-GNN has dramatically enhanced capabilities in exploring node features and graph structures simultaneously, and is suitable and adaptive for both feature-driven and structure-driven tasks. We also demonstrate that Eigen-GNN has several desirable theoretical properties such as permutation-equivariance and generality in plugging into many existing GNNs.

To assess the effectiveness of our proposed method, we conduct extensive experiments for tasks including node classification, link prediction, and graph isomorphism tests. The experimental results show that our proposed method consistently and significantly outperforms the baselines when the tasks and datasets are more structure-driven, and retains comparable performance with existing GNNs in feature-driven scenarios.

Our contributions are summarized as follows:

- We demonstrate that most existing shallow GNNs cannot preserve graph structures well in practice through both empirical analysis and analytical exploration.
- We propose Eigen-GNN, a simple yet effective and general plug-in module to boost GNNs ability in preserving graph structures. Eigen-GNN has several desirable theoretical properties and can be applied to many existing GNN architectures.
- Our extensive experimental results demonstrate that the proposed Eigen-GNN can preserve both features and graph structures more effectively and flexibly.

The rest of the paper is organized as follows. In Section 2, we review related works. In Section 3, we experimentally and analytically investigate whether the existing shallow GNNs can preserve graph structures well in practice. Our proposed method is introduced in Section 4, and experimental results are reported in Section 5. Finally, we conclude our work in Section 6.

2 Related Work

We briefly review related works of GNNs and refer readers to [12], [13], [14] for comprehensive surveys.

The earliest GNNs adopted recursive definitions of node states [15], [16] or a contextual realization [17]. More recently, Spectral GCNs [18] defined graph convolutions using graph signal processing [19]. ChebNet [20] and GCN [9] approximated the spectral graph convolution filters using a K-order Chebyshev polynomial and the first-order polynomial function, respectively. Duvenaud *et al.* [21] also considered the first-order neighborhood. MPNNs [1], GraphSAGE [22], MoNet [23] unified these methods using a "message-passing" framework, i.e., nodes aggregate information from neighborhoods as messages. Later studies such as GAT [24], JK-Nets [2], GIN [25], and GraphNets [26] usually follow these frameworks as more advanced variants.

1. The source codes are available at https://github.com/ZW-ZHANG/EigenGNN.

To understand the effectiveness of GNNs, Li *et al.* [7] showed that GNNs are a special form of Laplacian smoothing. Hou *et al.* [27] further proposed a metric to measure the smoothness of node features and node labels. Wu *et al.* [11] showed that the existing GNNs are equivalent to a fixed low-pass filter of graph signals and proposed an extremely simplified GNN by removing all the non-linearities. Maehara [10] took a similar idea and showed that adding an extra Multi-Layer Perceptron (MLP) layer can further increase the non-linear manifold learning capability of GNNs. Kipf and Welling [9], Zhang *et al.* [28], Xu *et al.* [25], Morris *et al.* [29], and Maron *et al.* [30] considered the connection between GNNs and the Weisfeiler-Lehman (WL) kernel for graph isomorphism tests. Dehmamy *et al.* [4] showed that GNNs with an infinite number of layers can preserve graph moments of any order and Loukas [6] established the lower bound for message-passing GNNs to calculate certain graph problems. However, whether shallow GNNs can preserve graph structures well in practice remains an open problem. P-GNN [31] proposed to preserve the position information of nodes by randomly selecting anchor nodes. However, since P-GNN only considers relative positions between nodes, it can only handle tasks for a pair of nodes such as link prediction and pairwise node classification, but not tasks for single nodes such as semi-supervised node classification or tasks for the whole graph such as graph classification.

How to design graph pooling methods while considering graph structures has also been studied [32], [33], [34]. In principle, our method can work jointly with them for graph-level tasks. Besides, there are recent attempts in increasing the depth of GNNs [35], [36], [37], [38], [39], which is also orthogonal to the study of this paper.

3 How Well Can Shallow GNNs Preserve Graph Structures?

In this section, we investigate whether shallow GNNs can preserve graph structures well in practice. We first report our observations from an empirical study. Then we obtain some insights into the findings from a dimensionality reduction perspective.

3.1 An Empirical Study

To manifest the capability of shallow GNNs in structure-driven and feature-driven tasks, we first conduct some experiments on synthetic datasets.

Datasets Generation and Methods in Comparison We generate synthetic datasets with two components: graph structures and node features. For graph structures, we partition nodes into l ($l > 0$) balanced communities and generate edges using the Stochastic Blockmodel [40], a representative method in generating community graphs. The nodes within the same community have a high probability of forming edges and those in different communities have a low probability of forming edges. We use the id of the community (a positive integer between 1 and l) that a node belongs to as the structure-driven label c_{struc} of the node.

For node features, we randomly divide nodes into l balanced groups. We generate a random vector for each

group, called the *group vector*. The features of a node are generated following a Gaussian distribution with the mean being the group vector of the group that the node belongs to. In this way, nodes within the same group share similar features. The group id (also a positive integer between 1 and l) is used as the feature-driven label c_{feat} of the node.

The final node label follows a Bernoulli distribution: $c = c_{\text{struc}}$ with probability γ and $c = c_{\text{feat}}$ with probability $1-\gamma$, where $0 \leq \gamma \leq 1$ is a parameter controlling the degree to which the node label prediction task is structure- or feature-driven. We call all the nodes carrying the same label as a *class*. Among all the nodes in class i $(1 \leq i \leq l)$, some are assigned the label due to the structure and the others are assigned the label due to and are manifested by the features. As two extremes, when $\gamma = 1$, the node label prediction task is completely structure-driven and, when $\gamma = 0$, the task is completely feature-driven. The larger the value of γ, the task is more structure-driven and less feature-driven. More details about the synthetic datasets can be found in Appendix B.1. We compare three different methods:

- GCNX_{feature}: this is GCN [9] taking features as inputs. Parameter X indicates the number of layers in the GCN and we test with 1, 2, 3, and 5 layers.
- MLP$_{\text{feature}}$: we use a two-layer Multi-Layer Perceptron on node features [9], i.e., a neural network with two fully connected layers. MLP$_{\text{feature}}$ does not learn any graph structure.
- DeepWalk [41]: a network embedding method to learn node representations and preserve graph structures. No node feature is used. We add a fully connected layer and a softmax layer on the learned embedding vectors for classification.

More experimental settings are provided in Appendix B.1. Note that we only adopt GCN, which is one of the most representative and widely-adopted GNN variants, in this empirical study as an illustration, while more GNNs are adopted and compared in the experiments (Section 5). Please also ignore the curves of Eigen-GCN in Figures 1a and 1b, which will be discussed later in Section 5.2.1.

Observations. First, we consider the extreme case where only graph structures are useful in the label prediction, i.e., $\gamma = 1$. In this case, to perform well, a model has to learn sufficient information about graph structures. Figure 1a shows the results. We have the following findings.

- The accuracy of MLP$_{\text{feature}}$ is about 10%. Since there are 10 balanced classes, this accuracy is roughly the same as random guessing. This verifies that node features indeed are not useful here.
- GCNs outperform MLP$_{\text{feature}}$, indicating that the existing GCNs can extract and exploit some information from the graph structures. These findings are consistent with the literature [9].
- Increasing the number of hidden layers in GCNs from 1 to 3 improves the accuracy. This verifies that deeper GCNs have a better capability in preserving graph structures. However, when GCNs have more layers, i.e., GCN5$_{\text{feature}}$, the performance tends to saturate or even drop (though residual connections are added), showing that training deep GCNs has unsolved practical challenges. The results are consistent with the literature [42].

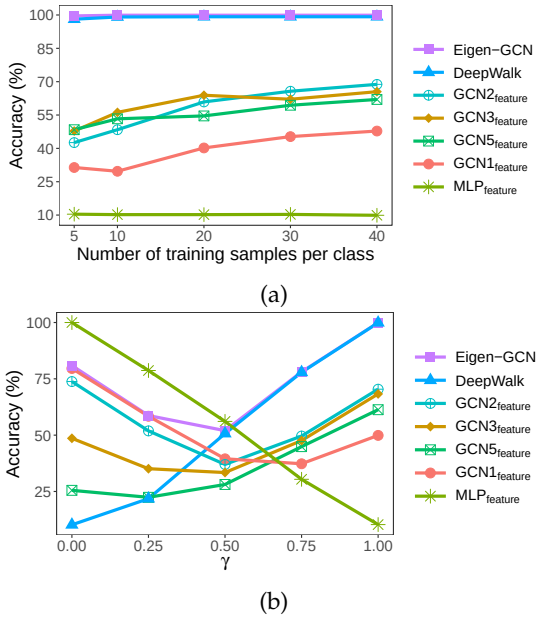


Fig. 1: The experimental results on synthetic datasets (a) in a completely structure-driven task, i.e., $\gamma = 1$, (b) when varying γ between 0 and 1. Best viewed in color.

- DeepWalk outperforms all the existing GCNs. This illustrates the weakness of GCNs in preserving graph structures. DeepWalk conducts random walks and takes the skip-gram model [43] to explicitly preserve graph structures. GCNs only utilize graph structures in aggregating node neighborhoods. The insufficiency of preserving graph structures in GCNs explains the inferior performance of GCNs in structure-driven tasks.

Next, we vary γ to mimic different kinds of tasks. Recall that the larger γ, the more structure-driven a task, and vice versa. The results are shown in Figure 1b. We have the following observations.

- When γ approaches 1, the results are consistent with those in Figure 1a. DeepWalk preserves graph structures better than GCNs. MLP$_{\text{feature}}$ gets the worst results since it does not use any graph structural information.
- When γ approaches 0, i.e., the task is heavily feature-driven, MLP$_{\text{feature}}$ achieves the best results. GCNs achieve inferior performance as they are misled to some extent by graph structures. DeepWalk performs poorly, the performance being similar to random guess, since it does not utilize any feature information.
- No existing method can perform well with respect to various γ values. This indicates that the existing models cannot preserve features and structures well simultaneously.

In summary, the experimental results on synthetic datasets illustrate that the existing shallow GNNs cannot preserve graph structures well in practice. Indeed, no existing method can be consistently competent in both structure- and feature-driven tasks. To better understand this phenomenon, we provide an analytical analysis using dimensionality reduction.

TABLE 1: GNN methods following Eq. (1) and their corresponding graph structure functions. $\mathbf{A}_P$ is the Positive Pointwise Mutual Information (PPMI) matrix [44].

Method	$\mathcal{F}(\mathbf{A})$
GCN [9]	$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$
SGC [11]	$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$
DCNN [45]	$(\mathbf{D}^{-1}\mathbf{A})^K$
DGCN [44]	$\mathbf{D}_P^{-\frac{1}{2}}\mathbf{A}_P\mathbf{D}_P^{-\frac{1}{2}}$
PPNP [3]	$\alpha(\mathbf{I}_N-(1-\alpha)\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})^{-1}$
MixHop [46]	$[\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}},...,(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})^j]$

3.2 GNNs as Dimensionality Reduction

Consider a graph $\boldsymbol{G}=(\mathcal{V},\mathcal{E},\mathbf{X})$, where $\mathcal{V}=\{v_1,...,v_N\}$ is a set of N nodes, $\mathcal{E}\subseteq\mathcal{V}\times\mathcal{V}$ is a set of M edges, and $\mathbf{X}\in\mathbb{R}^{N\times f}$ is an optional node feature matrix where f is the number of features. Denote by $\mathbf{A}\in\mathbb{R}^{N\times N}$ the adjacency matrix, and by $\mathbf{A}_{i,:}$, $\mathbf{A}_{:,j}$, and $\mathbf{A}_{i,j}$, respectively, the i^{th} row, the j^{th} column and an element in the matrix. We assume connected and undirected graphs, i.e., $\mathbf{A}_{i,j}=\mathbf{A}_{j,i}, 1\leq i,j\leq N$. We use bold uppercases (e.g., $\mathbf{Z}$) and bold lowercases (e.g., $\mathbf{z}$) to denote matrices and vectors, respectively. Functions are marked by curlicue, e.g., $\mathcal{F}(\cdot)$. We denote a non-linear activation function such as sigmoid or ReLU as $\sigma(\cdot)$.

Our analysis starts with the observation that many existing GNNs can be unified into the following framework. Denote by $\mathbf{H}^{(l)}\in\mathbb{R}^{N\times d_l}$ the representations of the nodes in the l^{th} hidden layer, where d_l is the dimensionality of layer l and $\mathbf{H}^{(0)}=\mathbf{X}$ are the input features, by $\mathbf{W}^{(l)}\in\mathbb{R}^{d_l\times d_{l+1}}$ the parameters, and by $\mathcal{F}(\mathbf{A})\in\mathbb{R}^{N\times N}$ a function on the graph structure. The l^{th} layer in a GNN is formulated as:

$$\mathbf{H}^{(l)}=\sigma(\mathcal{F}(\mathbf{A})\,\mathbf{H}^{(l-1)}\mathbf{W}^{(l-1)}). \tag{1}$$

For example, a well-known GNN variant, GCN [9] adopts the following function:

$$\mathcal{F}(\mathbf{A})=\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{2}$$

where $\tilde{\mathbf{A}}=\mathbf{A}+\mathbf{I}_N$, $\mathbf{I}_N$ is the $N\times N$ identity matrix, and $\tilde{\mathbf{D}}_{i,i}=\sum_j\tilde{\mathbf{A}}_{i,j}$ is the diagonal degree matrix. We list other five well-known GNNs following this framework such as DCNN [45] and PPNP [3], and their corresponding $\mathcal{F}(\mathbf{A})$ in Table 1.

Denote by $\mathbf{F}=\mathcal{F}(\mathbf{A})$. $\mathbf{F}$ encodes the raw structure information of the graph. For example, Eq. (2) shows that

$$\mathbf{F}_{i,j}=(\tilde{\mathbf{D}}_{i,i}\tilde{\mathbf{D}}_{j,j})^{-\frac{1}{2}}\tilde{\mathbf{A}}_{i,j}. \tag{3}$$

That is, $\mathbf{F}_{i,:}$ is a normalized adjacent vector of node v_i, encoding the second-order proximity between nodes [47]. In DCNN and PPNP, $\mathbf{F}$ encodes the transition probability between nodes. Eq. (1) can be interpreted as a three-step dimensionality reduction process by executing the calculation from left to right:

- Step 1: $\mathbf{F}'=\mathbf{F}\mathbf{H}^{(l)}$, i.e., projecting $\mathbf{F}\in\mathbb{R}^{N\times N}$ into a subspace spanned by $\mathbf{H}^{(l)}\in\mathbb{R}^{N\times d_l}$ to obtain a low-dimensional representation $\mathbf{F}'\in\mathbb{R}^{N\times d_l}$.
- Step 2: $\mathbf{F}'$ is further transformed by a linear mapping $\mathbf{W}^{(l)}\in\mathbb{R}^{d_l\times d_{l+1}}$ followed by a non-linear function $\sigma(\cdot)$, i.e., $\mathbf{H}^{(l+1)}=\sigma(\mathbf{F}'\mathbf{W}^{(l)})\in\mathbb{R}^{N\times d_{l+1}}$ as refined low-dimensional representations.
- Step 3: repeat the above two steps using $\mathbf{H}^{(l+1)}$ as the new base in Step 1.

Remark 1. GNNs can be regarded as a (non-linear) dimensionality reduction procedure with each layer performing one dimensionality reduction process. The node features provide the initial bases for the dimensionality reduction.

Now we can understand the inherent difficulty in the existing shallow GNNs to preserve graph structures. Specifically, the number of iterations in the dimensionality reduction is determined by the number of layers. Since most existing GNNs in practice are shallow, the initial bases play crucial roles and provide important inductive biases for GNNs. If the initial bases are solely determined by node features as in the existing GNNs, the resulted models are feature-centric and cannot well preserve graph structures.

In addition, the existing GNNs are struggling to handle the situations when no node feature is available. A commonly used trick is to use a one-hot encoding of node IDs [9], [48], i.e., $\mathbf{X}=\mathbf{I}_N$. However, using a one-hot encoding will dramatically increase the number of parameters and make the model unable to retain permutation-equivariance [48]. Another heuristic method is to use node degrees as node features [25], but it can only encode limited graph structure information.

4 EIGEN-GNN

Can we remedy the existing shallow GNNs in a principled way so that they can gain a strong capability of preserving graph structure information?

4.1 The Model

As analyzed in Section 3.2, the main reason that the existing shallow GNNs fail to preserve graph structures well is that the initial dimensionality reduction bases, $\mathbf{H}^{(0)}=\mathbf{X}$, are completely biased to features only and do not contain any structure information. To fix the problem, we need to find a suitable space where useful graph structure information can be preserved. It is well known in spectral graph theory [49] that the eigenspace of a graph provides informative low-dimensional spaces regarding graph structures. For example, spectral clustering [50] adopts the eigenvectors associated with the top-d smallest eigenvalues of the Laplacian matrix for node clustering, and network embedding adopts the eigenvectors associated with the top-d largest absolute eigenvectors of a polynomial function of the adjacency matrix for unsupervised node representation learning [51]. Inspired by those successes, our idea is to integrate the eigenspace of graph structures with GNNs by expanding the initial dimensionality reduction bases.

Consider a matrix $\mathcal{G}(\mathbf{A})\in\mathbb{R}^{N\times N}$ that encodes fruitful graph structure information. We aim to integrate the eigenspace of $\mathcal{G}(\mathbf{A})$ into GNNs. In this paper, we reuse the symmetrically normalized adjacency matrix in Eq. (2) as the graph structure matrix, i.e., $\mathcal{G}(\mathbf{A})=\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ since this matrix is widely used in GNNs, but our method can be

easily generalized to other matrices, such as the Laplacian matrix[2] or the transition matrix.

To keep it simple and general, we expand the initial dimensionality reduction bases by directly concatenating the eigenvectors of $\mathcal{G}(\mathbf{A})$ with node features

$$\mathbf{H}^{(0)} = [\mathbf{X}, f(\mathbf{Q})], \tag{4}$$

where $\mathbf{X}$ is the feature matrix, $\mathbf{Q} \in \mathbb{R}^{N\times d}$ are the eigenvectors corresponding to the top-d largest absolute eigenvalues of $\mathcal{G}(\mathbf{A})$, $f(\cdot)$ is a simple function such as normalization or identity mapping, and $[\cdot,\cdot]$ is the concatenation operator. Mathematically, Eq. (4) provides (potentially non-orthogonal) bases for the union space of the feature space and the eigenspace, and thus integrates these two spaces. In this paper, we have focused on concatenation because it is the most straight-forward mechanism to fuse the node feature space and the eigenspace, and we leave exploring more advanced methods, e.g., using gating or attention mechanism, as future works.

Rather than being a new GNN architecture, our proposed Eigen-GNN can be used as a plug-in module to enhance the capability of many existing GNNs in preserving graph structures. As both node features and graph structure information are captured in the initial dimensionality reduction bases, Eigen-GNN is flexible and adaptive in handling both structure-driven and feature-driven tasks since the dimensionality reduction process can freely explore these two spaces. Moreover, since the eigenspace is independent of node attributes, Eigen-GNN can easily handle featureless graphs by only using the eigenspace, as opposed to the existing GNNs that can only use heuristics such as node IDs or degrees.

Moreover, since Eigen-GNN only provides the initial dimensionality reduction bases, it can work jointly with different GNNs, including those designed for signed or multi-relational graphs, like propagating between positive/negative edges [52] and learning different weights for different edge types [53]. When generalizing to bipartite or directed graphs, we can simply replace the eigenvectors in Eq. (4) by singular vectors [54]. Besides, though we motivate our method using the framework in Eq. (1), our method is general enough to work with GNNs beyond this framework.

4.2 Several Desirable Properties of Eigen-GNN

We show that Eigen-GNN has several desired properties.

When applied to graph-level tasks such as graph classification [5], [55], a key property of the existing GNNs is permutation-equivariance, i.e., the node representations are equivariant if node IDs are permutated. Mathematically, permutation-equivariance reflects one basic symmetric group of graph structures. However, heuristics mentioned above such as one-hot IDs or using network embedding methods like Deepwalk cannot maintain this key property (unless we enumerate all possible permutations, which is exponential w.r.t. the number of nodes and not feasible for graphs with more than a dozen nodes; see [48], [56], [57]). To the contrary, we prove that one Eigen-GNN variant can maintain permutation-equivariance as long as the top-d eigenvalues of $\mathcal{G}(\mathbf{A})$ are unique.

Theorem 1. For two graphs $\boldsymbol{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ and $\boldsymbol{G}' = (\mathcal{V}', \mathcal{E}', \mathbf{X}')$, we assume the top-$d$ eigenvalues of $\mathcal{G}(\mathbf{A})$ are unique for $\boldsymbol{G}$ and $\boldsymbol{G}'$ and use $f(x) = |x|$ in Eq. (4). Then, Eigen-GNN is permutation-equivariant if the base GNN follows Eq. (1). Specifically, we denote by $\mathbf{H}^{(l)}, \mathbf{U}^{(l)}, 0 \le l \le L$, respectively, the representations of $\mathcal{V}$ and $\mathcal{V}'$ in the l^{th} hidden layer of the Eigen-GNN. If there exists a bijective mapping $\mathcal{B} : \mathcal{V} \to \mathcal{V}'$ so that $\mathcal{E}(i,j) = \mathcal{E}'(\mathcal{B}(i), \mathcal{B}(j)), \mathbf{X}_{i,:} = \mathbf{X}'_{\mathcal{B}(i),:}, \forall 1 \le i,j \le N$, then, $\mathbf{H}^{(l)}_{i,:} = \mathbf{U}^{(l)}_{\mathcal{B}(i),:}, \forall 1 \le i \le N, \forall 0 \le l \le L$.

The proof is given in Appendix C.1. By being able to maintain permutation-equivariance, Eigen-GNN can be applied to graph-level tasks [3]. We further demonstrate this advantage empirically in Section 5.3. Though we only prove the case for graphs with unique top eigenvalues, our experiments in Section 5.3 adopt regular graphs, which contain non-unique top eigenvalues. Empirical results show that Eigen-GNN works reasonably well in those cases. We leave theoretical analysis for non-unique eigenvalues as future works.

In addition, Eigen-GNN is scalable to large graphs since we only calculate the eigenvectors corresponding to the largest absolute eigenvalues. We have the following result.

Remark 2. The time complexity of calculating the eigenspace in Eq. (4) is $O\left(T\left(M_{\mathcal{G}}d + Nd^2\right)\right)$, where $M_{\mathcal{G}}$ is the number of non-zero elements in $\mathcal{G}(\mathbf{A})$, N is the number of nodes, d is the preset dimensionality, and T is the number of iterations (a constant).

Proof. The result is due to well-known iterative algorithms for calculating the eigenspace in linear algebra such as the Arnoldi method [59]. □

The result shows that the time complexity mainly depends on the number of non-zero elements in the graph structure matrix $\mathcal{G}(\mathbf{A})$. By setting $\mathcal{G}(\mathbf{A})$ as a sparse matrix, e.g., the normalized adjacency matrix, we have $M_{\mathcal{G}} \approx M$. In such a case, the time complexity of calculating the eigenspace is linear with respect to the number of nodes and that of edges in the graph. Since this time complexity is on the same scale as the existing GNNs, Eigen-GNN does not incur any extra cost in scalability. We empirically verify the result in Section 5.5 by showing that we can handle graphs with tens of thousands of nodes and millions of edges in a few seconds using a normal server. Notice that unlike spectral GCNs [18], we only utilize the eigenvectors associated with the top eigenvalues rather than the full spectrum, so that our algorithm is much more scalable.

Finally, we show an interesting connection between our method and Simple Graph Convolution (SGC) [11], a simplified GNN variant without non-linearities.

2. Recall that the eigenspace of the normalized adjacency matrix and the normalized Laplacian matrix encode similar information. Specifically, if $\mathbf{x}$ is an eigenvector of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ with eigenvalue λ, $\mathbf{x}$ is also an eigenvector of $\tilde{\mathbf{L}}_{sym} = \tilde{\mathbf{D}}^{-\frac{1}{2}}(\tilde{\mathbf{D}} - \tilde{\mathbf{A}})\tilde{\mathbf{D}}^{-\frac{1}{2}}$ with eigenvalue $1-\lambda$, and vice versa.

3. Notice that we only adopt the permutation-equivariant Eigen-GNN variant, i.e., $f(x) = |x|$, for graph-level tasks since permutation-equivariance is strongly required. For node- or edge-level tasks, it is shown in [31], [58] that permutation-equivariance may not be a desirable property, thus we adopt normal Eigen-GNNs, e.g., $f(x) = x$.

Theorem 2. For a graph that is not bipartite, an SGC with an infinite number of layers converges to Eigen-GNN with no hidden layer and the eigenspace dimensionality $d = 1$.

The proof is given in Appendix C.2. In fact, we can generalize this result to GCNs with ReLU activation functions under mild assumptions [8]. The theorem implies that, instead of integrating graph structures gradually in each layer as an SGC, Eigen-GNN can directly provide the final graph structure information used by SGC using a "shortcut" by the first eigenspace and thus better preserve graph structures without needing to increase depths.

5 Experimental Results

Since Eigen-GNN is a general plug-in to enhance existing GNNs rather than a new architecture, we conduct a series of experiments to answer the following three questions.

- **Q1**: Can Eigen-GNN improve GNNs in structure-driven tasks? Does Eigen-GNN impair feature-driven tasks?
- **Q2**: Can Eigen-GNN be easily plugged into various GNN models?
- **Q3**: Can we empirically verify the desirable properties of Eigen-GNN in applications?

5.1 Baselines and Experimental Settings

We compare the following three methods:

- GNN_{feat}: we report the original results of the GNN model with node features as inputs.
- $\text{GNN}_{\text{feat+DW}}$: we run DeepWalk [41] on graph structures and concatenate the generated embedding vectors with node features as inputs to GNNs. This is a heuristic approach to enhance GNNs in preserving graph structures [60].
- Eigen-$\text{GNN}_{\text{feat+struc}}$: our proposed method, i.e., we concatenate the eigenspace with node features as inputs.

We also include five methods without using node features.

- $\text{GNN}_{\text{one-hot}}$: we use a one-hot encoding of node ID as inputs to GNNs [48].
- $\text{GNN}_{\text{degree}}$: we use a one-hot encoding of node degrees as inputs to GNNs, which is proven useful in chemistry graphs [25].
- $\text{GNN}_{\text{random}}$: we generate random features following a Gaussian distribution as inputs to GNNs [61].
- GNN_{DW}: we use the embedding vectors of DeepWalk as inputs to GNNs.
- Eigen-$\text{GNN}_{\text{struc}}$: we adopt the eigenspace as the inputs.

Although Eigen-GNN can be generally plugged into many different GNNs, it is infeasible to compare every possible GNN architecture due to the vast and fast-developing literature. Instead, we adopt the most prominent GNNs for the tasks as showcases (the exact model will be given in each subsection). We further clarify the adopted architecture by replacing the "GNN" in method names with the exact model name, e.g., Eigen-GCN if we use GCN and Eigen-GAT if we use GAT. For hyper-parameters, we search the dimensionality d of the eigenspace from $\{32, 64, 128, 256\}$. We repeat all experiments 10 times and report the average results and standard deviation of different runs. Additional hyper-parameters and details for reproducibility can be found in Appendix B.

5.2 Node Classification

5.2.1 Revisiting the Empirical Study in Section 3.1

Section 3.1 presents an empirical study using synthetic datasets. Now let us examine the performance of Eigen-GCN (we use GCN as the base GNN architecture) in Figures 1a and 1b.

- When the task is structure-driven (i.e., γ approaches 1), Eigen-GCN achieves the best performance. This shows that our plugged-in module can empower GCNs to better capture graph structure information.
- Eigen-GCN achieves the most stable performance with respect to γ varying between 0 and 1. This demonstrates that Eigen-GCN can handle both feature-driven and structure-driven tasks. Eigen-GCN consistently outperforms the existing GCNs when $\gamma \geq 0.5$ and retains comparable results when $\gamma < 0.5$, showing that Eigen-GCN is robust and thus a reliable choice even when the type of the tasks is unknown.
- When the task is feature-driven (i.e., γ approaches 0), although Eigen-GCN outperforms GCN, $\text{MLP}_{\text{feature}}$ reports better results, showing that a graph-based method may not be preferred in those cases after all.

5.2.2 Results on Real-world Datasets

We further experiment on 7 real-world social networks [62]: Harvard, Columbia, Stanford, Yale, Cornell, Dartmouth, and UPenn [62][4]. These are Facebook social networks for different colleges/universities. Edges represent intra-school links of users and node attributes correspond to user profiles such as gender, major, dorm/house, etc. We use the class year as ground-truth labels. We preprocess the datasets by using a one-hot encoding of categorical node features and removing node features/labels which occur less than 0.1%/1% among all the nodes.

The statistics of the datasets are summarized in Table 3. For the Facebook social networks, we use 20 nodes per class for training, 30 nodes per class for validation, and the rest for testing. For the other four benchmark datasets, we adopt the fixed training/validation/testing split that came with the datasets. Similar results are observed in random splits.

For the base GNN model, we adopt three widely used architectures: GCN [9], GAT [24], and GraphSAGE [22]. We report the results of using GCN in Table 2. The results of using GAT and GraphSAGE show a similar trend and are provided in Appendix A.1 due to the page limit. We also omit the results of $\text{GNN}_{\text{one-hot}}$ since it runs out of memory on most of the datasets. We make the following observations.

- Eigen-$\text{GCN}_{\text{feat+struc}}$ reports the best results on all the datasets and the improvements of Eigen-$\text{GCN}_{\text{feat+struc}}$ compared to Eigen-GCN_{feat} are more than 10% in terms of the classification accuracy on 4 datasets (Harvard, Yale, Dartmouth, and UPenn). The results clearly demonstrate that graph structures are crucial in this task and our proposed method can greatly enhance the existing GCNs in preserving graph structures.
- When node features are unavailable, Eigen-$\text{GCN}_{\text{struc}}$ also consistently outperforms the other methods. This

4. https://archive.org/details/oxford-2005-facebook-matrix

TABLE 2: The accuracy (%) of node classification on 7 social networks using GCN as the base model (results using GAT and GraphSAGE as the base model show similar trends and are provided in Appendix A.1). The best results with and without node features, respectively, are in bold. A, X, Y stands for graph structures, node features, and node labels, respectively.

Data	Method	Harvard	Columbia	Stanford	Yale	Cornell	Dartmouth	UPenn
A,Y	$\text{GCN}_{\text{random}}$	74.6 ± 0.5	63.6 ± 1.6	68.2 ± 1.5	73.6 ± 1.2	54.5 ± 1.7	73.1 ± 1.6	63.0 ± 1.2
	$\text{GCN}_{\text{degree}}$	74.4 ± 2.0	63.8 ± 2.3	67.8 ± 1.6	76.5 ± 2.0	56.3 ± 1.6	73.3 ± 1.4	65.4 ± 1.8
	GCN_{DW}	82.5 ± 1.0	$\mathbf{76.0 \pm 1.3}$	76.6 ± 1.3	82.6 ± 1.0	71.0 ± 2.0	79.3 ± 1.7	77.1 ± 1.4
	$\text{Eigen-GCN}_{\text{struc}}$	$\mathbf{82.7 \pm 1.2}$	$\mathbf{76.0 \pm 1.9}$	$\mathbf{78.9 \pm 1.3}$	$\mathbf{84.2 \pm 1.4}$	$\mathbf{71.9 \pm 1.7}$	$\mathbf{82.1 \pm 1.2}$	$\mathbf{78.5 \pm 1.4}$
A,X,Y	GCN_{feat}	70.6 ± 1.3	74.8 ± 1.7	71.3 ± 1.6	71.2 ± 1.9	67.0 ± 1.9	73.1 ± 1.6	71.2 ± 2.0
	$\text{GCN}_{\text{feat+DW}}$	83.1 ± 0.7	77.6 ± 1.3	78.3 ± 1.4	83.5 ± 1.5	73.2 ± 2.2	80.8 ± 1.3	78.5 ± 1.2
	$\text{Eigen-GCN}_{\text{feat+struc}}$	$\mathbf{84.6 \pm 1.4}$	$\mathbf{78.6 \pm 1.1}$	$\mathbf{79.7 \pm 1.2}$	$\mathbf{85.1 \pm 1.3}$	$\mathbf{74.8 \pm 1.8}$	$\mathbf{83.6 \pm 1.3}$	$\mathbf{81.3 \pm 0.9}$

TABLE 3: Statistics of the datasets for node classification.

Dataset	Type	#Nodes	#Edges	#Classes	#Features
Harvard	Social	15,126	1,649,234	10	136
Columbia	Social	11,770	888,666	7	197
Stanford	Social	11,621	1,136,660	8	225
Yale	Social	8,578	810,900	8	146
Cornell	Social	18,660	1,581,554	7	253
Dartmouth	Social	7,694	608,152	9	178
UPenn	Social	14,916	1,373,002	7	204
Cora	Citation	2,708	5,429	7	1,433
Citeseer	Citation	3,327	4,732	6	3,703
Pubmed	Citation	19,717	44,338	3	500
Reddit	Social	232,965	11,606,919	41	602

demonstrates that Eigen-GCN can extract fruitful information from the graph structures and handle featureless graphs.

- GCN_{DW} and $\text{GCN}_{\text{feat+DW}}$ achieve the second-best results. This heuristic method works reasonably well, but is still inferior to Eigen-GCN.

5.2.3 Results on Benchmarks

We also experiment on four benchmark datasets commonly used in GNNs.

- Cora, Citeseer,Pubmed [63][5]: citation graphs where nodes represent papers and edges represent citations between papers. The datasets also contain bag-of-words features and ground-truth topics as labels of the papers.
- Reddit [22][6]: an online discussion forum for users where nodes are posts and two nodes are connected if they are commented by the same user. Each post contains a low-dimensional word vector as features. The task is to predict which community the posts belong to.

The results are shown in Table 4. For simplicity, we only adopt GCN [9] as the base GNN model. We make the following observations.

- Similar to Section 5.2.2, when no node feature is available, $\text{Eigen-GCN}_{\text{struc}}$ reports the best results, demonstrating that Eigen-GCN can better preserve graph structures.
- When features are available, GCN_{feat} performs the best on three citation graphs and highly competently on Reddit, showing that features are dominant on these benchmark datasets. The results are consistent with the literature [10], [11], which show that features contain the "true signals" for those node classification tasks.
- $\text{Eigen-GCN}_{\text{feat+struc}}$ has comparable performance with GCN_{feat} on the three citations graphs and is even better than GCN_{feat} on Reddit. These results show that expanding the initial bases with the eigenspace does not impair GNNs in feature-driven tasks. Thus, Eigen-GNN can be adopted as a default module if we are not sure whether a task is feature-driven or structure-driven.

5. https://github.com/tkipf/gcn
6. http://snap.stanford.edu/graphsage/

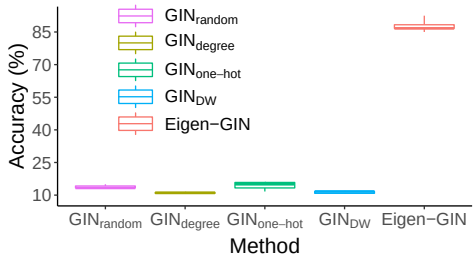


Fig. 2: The results of graph isomorphism tests on circulant skip link graphs.

5.2.4 Summary

The experimental results demonstrate that Eigen-GNN achieves superior performance on structure-driven tasks and does not affect the performance when node features are dominant. Moreover, Eigen-GNN can handle featureless graphs well.

5.3 Graph Isomorphism Tests

We further conduct experiments on Circulant Skip Links (CSL) graphs [48], a well-known dataset for graph isomorphism tests, i.e., distinguishing whether two graphs are structurally equivalent. We briefly introduce CSL graphs as follows. A basic CSL graph $\boldsymbol{G}_{N,R}$ is an undirected graph, where $\{1, ..., N\}$ is the set of N nodes and the edges consist of a cycle and a set of skip links. We denote $\mathbf{A}$ as the adjacency matrix. The cycle is formulated as:

$$\mathbf{A}_{j,j+1} = \mathbf{A}_{j+1,j} = 1, \forall 1 \leq j < N \tag{5}$$

$$\mathbf{A}_{1,N} = \mathbf{A}_{N,1} = 1. \tag{6}$$

The skip links, controlled by an interval parameter R satisfying $1 < R < N$, are defined as:

$$\mathbf{A}_{i,j} = \mathbf{A}_{j,i} = 1, \text{if } |j - i| = R \text{ or } N - R \bmod N, \forall 1 \leq i, j \leq N. \tag{7}$$

TABLE 4: The results of node classification accuracy (%) on benchmark datasets. The best results with and without node features, respectively, are in bold. A, X, Y stands for graph structures, node features, and node labels, respectively.

Data	Method	Cora	Citeseer	Pubmed	Reddit
A,Y	$\text{GCN}_{\text{random}}$	23.5 ± 1.6	21.2 ± 1.1	32.6 ± 1.0	86.1 ± 0.3
	$\text{GCN}_{\text{degree}}$	33.5 ± 2.4	30.2 ± 0.9	34.9 ± 1.3	83.0 ± 0.4
	$\text{GCN}_{\text{one-hot}}$	66.3 ± 0.6	45.2 ± 1.1	64.3 ± 0.9	Out of memory
	GCN_{DW}	70.6 ± 1.2	47.7 ± 1.1	69.3 ± 1.2	$\mathbf{94.3 \pm 0.1}$
	$\text{Eigen-GCN}_{\text{struc}}$	$\mathbf{71.0 \pm 0.5}$	$\mathbf{49.3 \pm 0.6}$	$\mathbf{73.8 \pm 0.3}$	$\mathbf{94.3 \pm 0.0}$
A,X,Y	GCN_{feat}	$\mathbf{81.5 \pm 0.4}$	$\mathbf{70.6 \pm 0.8}$	$\mathbf{78.6 \pm 0.4}$	96.4 ± 0.0
	$\text{GCN}_{\text{feat+DW}}$	76.8 ± 0.5	61.8 ± 0.6	76.3 ± 0.5	$\mathbf{96.6 \pm 0.1}$
	$\text{Eigen-GCN}_{\text{feat+struc}}$	78.9 ± 0.7	66.5 ± 0.3	$\mathbf{78.6 \pm 0.1}$	$\mathbf{96.6 \pm 0.1}$

TABLE 5: The average precision of link prediction (%). The best results are highlighted in bold.

Dataset	C.elegans	E.coli	NS	PB	Power	Router	USAir	Yeast
SEAL	77.6 ± 0.9	91.5 ± 0.8	96.8 ± 1.8	87.3 ± 0.3	69.9 ± 1.6	88.2 ± 1.0	$\mathbf{90.3 \pm 1.6}$	$\mathbf{93.7 \pm 0.3}$
SEAL_{DW}	77.1 ± 1.5	92.1 ± 0.7	97.0 ± 1.2	87.5 ± 0.2	69.8 ± 1.5	87.9 ± 1.3	89.9 ± 1.9	93.6 ± 0.5
Eigen-SEAL	$\mathbf{79.5 \pm 0.8}^*$	$\mathbf{92.5 \pm 0.6}^*$	$\mathbf{97.2 \pm 0.6}$	$\mathbf{87.8 \pm 0.4}^*$	$\mathbf{73.2 \pm 2.4}^*$	$\mathbf{88.3 \pm 1.2}$	$\mathbf{90.3 \pm 1.2}$	93.6 ± 0.4
Gain†	$+1.9$	$+0.4$	$+0.2$	$+0.3$	$+3.3$	$+0.1$	0.0	-0.1

†: Gain is the relative improvement of Eigen-SEAL compared to the better of the other two methods.
*: The improvement of bolded results over non-bolded results is statistically significant at 0.05-level paired t-test.

Figure 3 shows examples of $\boldsymbol{G}_{13,2}$ and $\boldsymbol{G}_{13,3}$, i.e., two CSL graphs with 13 nodes and with skip links of the interval 2 and 3, respectively. Intuitively, basic CSL graphs $\boldsymbol{G}_{N,R}$ are 4-regular graphs (i.e., the degree of all nodes is 4) by connecting every "adjacent" node pair and every node pair that is "R-hops" away. The full CSL graph set includes the basic CSL graphs $\boldsymbol{G}_{N,R}$ and all their permutations, i.e.,

$$\boldsymbol{G}_N = \{\mathcal{S}_N\left(\boldsymbol{G}_{N,R}\right), \forall 1 < R < N, \forall \mathcal{S}_N\}, \tag{8}$$

where $\mathcal{S}_N(\cdot)$ is any permutation of N node IDs.

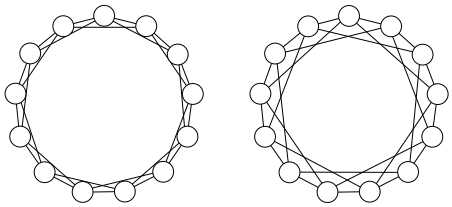


Fig. 3: An example of CSL graphs $\boldsymbol{G}_{13,2}$ and $\boldsymbol{G}_{13,3}$. Though these two graphs are non-isomorphism, their structures are extremely similar that the existing GNNs fail to distinguish them. The image is adapted from [48].

CSL graphs are widely adopted for graph isomorphism tests since their structures are highly regular and similar, and all nodes have the same degree. For example, it is known that $\boldsymbol{G}_{41}$ is composed of 10 isomorphism classes:

$$\boldsymbol{G}_{41} = \{\mathcal{S}_{41}\left(\boldsymbol{G}_{41,R}\right) | R \in \{2,3,4,5,6,9,11,12,13,16\}\}.$$

Although there exist known mathematical approaches to solve graph isomorphism tests for CSL graphs [64], it still poses great challenges for machine learning models, including the existing GNNs, to distinguish them if no prior knowledge is used [48], [65].

Specifically, following the experimental setting in [48], we consider the aforementioned CSL graphs with 41 nodes and 10 isomorphism classes. Using the isomorphism classes as labels for graphs, the graph isomorphism test can be transformed into a graph classification problem. For each isomorphism class, i.e., a graph label, we randomly generate 60 isomorphic CSL graphs belonging to that class. As a result, the dataset contains 600 graphs with 10 balanced classes. Following [48], we adopt a 5-fold cross-validation.

We adopt GIN [25] as the baseline GNN model, which is proven to be one of the most powerful message-passing GNN models in graph isomorphism tests[7]. Since this dataset does not contain node features, we only report in Fig 2 the results of the five methods that do not use node features. We make the following observations.

All the methods except for Eigen-GIN report an accuracy of about 10%, roughly the same as that of random guessing since the dataset has 10 balanced classes. These results are consistent with the theoretical findings that the original GIN (as well as other message-passing GNNs) cannot distinguish CSL graphs [48]. The major reason is that $\text{GIN}_{\text{random}}$, $\text{GIN}_{\text{one-hot}}$, and GIN_{DW} do not satisfy permutation-equivariance, a necessary requirement for graph isomorphism tests, and $\text{GIN}_{\text{degree}}$ cannot distinguish graph structures if nodes have the same degree.

Eigen-GIN reports a remarkably high accuracy. It can recognize CSL graphs well due to two reasons. First, Eigen-GIN satisfies permutation-equivariance, as proven in Theorem 1. Second, the eigenspace provides more fruitful structural information than simple heuristics such as degrees.

We also conduct graph isomorphism tests on random regular graphs. Specifically, we generate 10 non-isomorphic 4-regular graphs containing 20 nodes as isomorphism classes. Other experimental settings are kept the same as on CSL graphs. The results are shown in Figure 4. As on CSL graphs, our proposed Eigen-GIN greatly outperforms other baselines.

7. We do not adopt a more recent approach RP-GIN [48] because of its high time complexity.

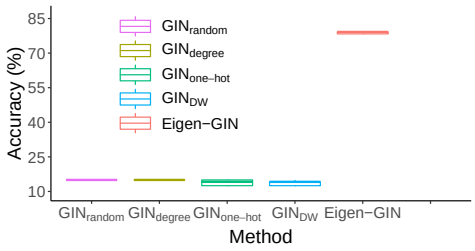


Fig. 4: The results of graph isomorphism tests on random regular graphs.

5.4 Link Prediction

Link prediction is to predict which pairs of nodes in a graph are most likely to form edges, which also involves graph structure information substantially. We adopt eight benchmark datasets from [66][8]:

- C.elegans: the neural network of the worm C.elegans.
- E.coli: a pairwise metabolites reaction network in E.coli.
- NS: a collaboration network between researchers, where nodes represent authors and edges correspond to co-authorships.
- PB: a graph formed by US political blogs where edges represent hyperlinks between blogs.
- Power: an electrical grid of the western US, where edges represent high-voltage transmission lines.
- Router: a router-level Internet connection graph.
- USAir: a graph from US Airlines with nodes representing airports and edges representing airlines.
- Yeast: a protein-protein interaction network in yeast.

TABLE 6: Statistics of the datasets for link prediction.

Dataset	Type	#Nodes	#Edges	Degree
C.elegans	Biology	297	4,296	14.5
Ecoli	Biology	1,805	29,320	16.2
NS	Collaboration	1,589	5,484	3.5
PB	Social	1,222	33,428	27.4
Power	Industry	4,941	13,188	2.7
Router	Internet	5,022	12,516	2.5
USAir	Transportation	332	4,252	12.8
Yeast	Biology	2,375	23,386	9.9

The statistics of the datasets are summarized in Table 6. Following [66], we randomly split the edges of the graph into 50%-20%-30% parts, and use them for training, validation, and testing, respectively. In splitting the datasets, we maintain that each node has at least one edge in the training set. The same number of edges are sampled from the non-existing links (i.e., node pairs that do not have edges) as negative samples.

We use SEAL [66] as the baseline GNN model, a state-of-the-art GNN specifically designed for link prediction. The architecture is kept the same as the original paper.

The results are reported in Table 5. We exclude the five baselines that do not use node features since SEAL has specifically designed those features and cannot function without them. We make the following observations.

8. https://github.com/muhanzhang/SEAL

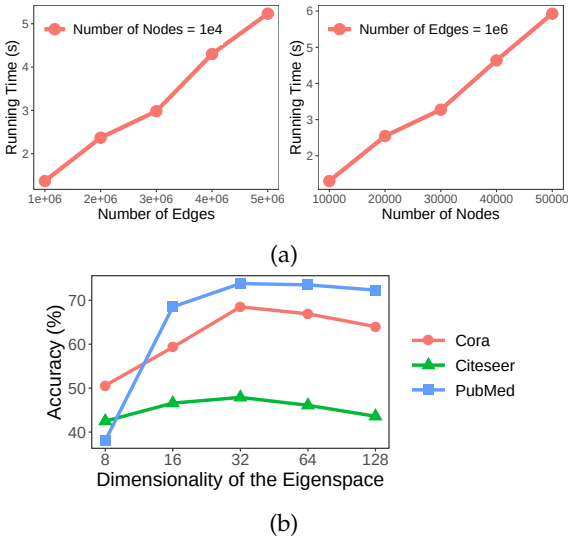


Fig. 5: Scalability and parameter sensitivity. (a) The running time of calculating the eigenspace grows linearly with respect to the number of nodes and the number of edges in the graph, respectively. (b) The node classification accuracy of Eigen-GNN with different eigenspace dimensionality.

- Eigen-SEAL reports significantly better results than the two baselines on four out of the eight datasets (with the rest four datasets showing no significant differences). This indicates that graph structures are important in link prediction tasks. The findings are consistent with the literature [67].
- Although SEAL is specifically designed for link prediction and Eigen-GNN does not target at any specific task, Eigen-SEAL reports better performance. This demonstrates the general effectiveness of Eigen-GNN.
- $\text{SEAL}_{\text{feat+DW}}$ fails to improve SEAL on any dataset[9]. This shows that DeepWalk cannot enhance SEAL in link prediction. The results are consistent with the paper [66].

5.5 Scalability and Parameter Sensitivity

5.5.1 *Scalability*

Since Eigen-GNN conducts the same calculation as base GNN models in all the hidden layers, we only report the runtime of calculating the eigenspace, which is the extra cost caused by Eigen-GNN. Specifically, we generate random graphs of different sizes using the Erdos Renyi model [68]. Figure 5a shows the runtime when fixing either the number of nodes to 10 thousand or fixing the number of edges to 1 million, and varying the other factor. The time of calculating the eigenspace increases roughly linearly with respect to the number of nodes and the number of edges in graphs. In addition, even for a large graph with 50 thousand nodes and 1 million edges, the running time is no more than 6 seconds on a single server. The results show that Eigen-GNN is scalable to large graphs.

9. Though $\text{SEAL}_{\text{feat+DW}}$ seems to perform better on E.coli, NS, and PB, the improvement is not statistically significant under 0.05 paired t-test.

5.5.2 Parameter Sensitivity

Eigen-GNN has only one parameter, the dimensionality d of the eigenspace. To test the parameter sensitivity, we follow the same experimental setting in Section 5.2.2 by adopting GCN [9] as the base GNN model and vary d in $\{8, 16, 32, 64, 128\}$. Figure 5b shows the node classification results on the three citations graphs without using node features. The results of the other tasks on the corresponding datasets share similar patterns. When the dimensionality d increases, the accuracy of the model increases at first but tends to saturate or even decreases if d becomes too large. A plausible reason is that, if the dimensionality of the eigenspace is too small, the model does not have enough capacities to learn useful graph structures. If the eigenspace grows too large, noises are likely to be introduced.

5.6 Summary

In summary, the experimental results show that Eigen-GNN works well with a number of GNN models for different tasks, namely GCN [9], GAT [24], GraphSAGE [22] for node classification, GIN [25] for graph isomorphism tests, and SEAL [66] for link prediction. These results well demonstrate the general applicability of Eigen-GNN in enhancing various GNNs in preserving graph structures.

6 Conclusion

In this paper, we observe that though GNNs with an infinite number of layers can preserve graph structures in theory, many GNNs in practice are shallow in nature and do not have a sufficient capability to well preserve graph structure. Then, motivated by treating GNNs as a type of dimensionality reduction, we propose Eigen-GNN, a simple yet general and effective plug-in module that integrates the eigenspace of graph structures with GNNs. We show that Eigen-GNN is capable of handling both feature-driven and structure-driven tasks simultaneously. Our extensive experiments demonstrate the effectiveness of Eigen-GNN in a wide spectrum of tasks including node classification, link prediction, and graph isomorphism tests.

Acknowledgments

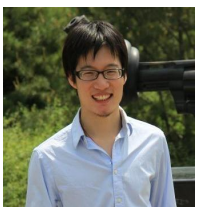
This work was supported in part by National Key Research and Development Program of China (No. 2020AAA0106300), National Natural Science Foundation of China (No. U1936219, No. 61521002, No. 61772304, No. 62050110, No. 62102222), and Beijing Academy of Artificial Intelligence (BAAI). All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. Peng Cui, Xin Wang, and Wenwu Zhu are corresponding authors.

References

[1] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*, 2017.
[2] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *ICML*, 2018.
[3] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *ICLR*, 2019.
[4] N. Dehmamy, A.-L. Barabási, and R. Yu, "Understanding the representation power of graph neural networks in learning graph topology," in *NeurIPS*, 2019.
[5] N. Keriven and G. Peyré, "Universal invariant and equivariant graph neural networks," in *NeurIPS*, 2019.
[6] A. Loukas, "What graph neural networks cannot learn: depth vs width," in *ICLR*, 2020.
[7] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018.
[8] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *ICLR*, 2020.
[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[10] T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," *arXiv:1905.09550*, 2019.
[11] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *ICML*, 2019.
[12] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *TNNLS*, 2020.
[13] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *TKDE*, 2020.
[14] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv:1812.08434*, 2018.
[15] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, 2009.
[16] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *IJCNN*, 2005.
[17] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, 2009.
[18] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *ICLR*, 2014.
[19] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, 2013.
[20] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016.
[21] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *NeurIPS*, 2015.
[22] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.
[23] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *CVPR*, 2017.
[24] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
[25] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*, 2019.
[26] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv:1806.01261*, 2018.
[27] Y. Hou, J. Zhang, J. Cheng, K. Ma, R. T. Ma, H. Chen, and M. Yang, "Measuring and improving the use of graph information in graph neural networks," in *ICLR*, 2020.
[28] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *AAAI*, 2018.
[29] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *AAAI*, 2019.
[30] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman, "Provably powerful graph networks," in *NeurIPS*, 2019.
[31] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *ICML*, 2019.
[32] E. Ranjan, S. Sanyal, and P. Talukdar, "Asap: Adaptive structure aware pooling for learning hierarchical graph representations," *AAAI*, 2020.
[33] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *KDD*, 2019.

[34] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, and C. Wang, "Hierarchical graph pooling with structure learning," *AAAI*, 2020.
[35] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *ICCV*, 2019.
[36] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020.
[37] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *ICLR*, 2020.
[38] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *KDD*, 2020.
[39] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *ICML*, 2020.
[40] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *JMLR*, 2008.
[41] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014.
[42] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE TSP*, 2018.
[43] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.
[44] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *WWW*, 2018.
[45] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *NIPS*, 2016.
[46] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *ICML*, 2019.
[47] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016.
[48] R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, "Relational pooling for graph representations," in *ICML*, 2019.
[49] F. R. Chung and F. C. Graham, *Spectral graph theory*, 1997.
[50] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *NIPS*, 2002.
[51] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, "Arbitrary-order proximity preserved network embedding," in *KDD*, 2018.
[52] T. Derr, Y. Ma, and J. Tang, "Signed graph convolutional networks," in *ICDM*, 2018.
[53] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*, 2018.
[54] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear algebra*. Springer, 1971.
[55] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman, "Invariant and equivariant graph networks," in *ICLR*, 2019.
[56] G. Dasoulas, L. D. Santos, K. Scaman, and A. Virmaux, "Coloring graph neural networks for node disambiguation," in *IJCAI*, 2020.
[57] B. Srinivasan and B. Ribeiro, "On the equivalence between positional node embeddings and structural graph representations," in *ICLR*, 2020.
[58] Z. Zhang, C. Niu, P. Cui, B. Zhang, W. Cui, and W. Zhu, "A simple and general graph neural network with stochastic message passing," *arXiv:2009.02562*, 2020.
[59] R. B. Lehoucq and D. C. Sorensen, "Deflation techniques for an implicitly restarted arnoldi iteration," *SIAM Journal on Matrix Analysis and Applications*, 1996.
[60] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "Deepinf: Modeling influence locality in large social networks," in *KDD*, 2018.
[61] R. Sato, M. Yamada, and H. Kashima, "Random features strengthen graph neural networks," *arXiv:2002.03155*, 2020.
[62] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of facebook networks," *Physica A: Statistical Mechanics and its Applications*, 2012.
[63] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, 2008.
[64] M. Muzychuk, "A solution of the isomorphism problem for circulant graphs," *Proceedings of the London Mathematical Society*, 2004.
[65] Z. Chen, S. Villar, L. Chen, and J. Bruna, "On the equivalence between graph isomorphism testing and function approximation with gnns," in *NeurIPS*, 2019.
[66] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *NeurIPS*, 2018.
[67] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, 2007.
[68] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, 1960.
[69] Z. Li, J. Tang, L. Zhang, and J. Yang, "Weakly-supervised semantic guided hashing for social image retrieval." *IJCV*, 2020.
[70] Z. Li, J. Tang, and T. Mei, "Deep collaborative embedding for social image understanding," *TPAMI*, 2018.
[71] Z. Li and J. Tang, "Semi-supervised local feature selection for data classification," in *Science China Information Sciences*, 2021.
[72] L. Babai, D. Y. Grigoryev, and D. M. Mount, "Isomorphism of graphs with bounded eigenvalue multiplicity," in *STOC*, 1982.
[73] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," in *ICML*, 2018.

Ziwei Zhang received his Ph.D. from the Department of Computer Science and Technology, Tsinghua University, in 2021. He is currently a postdoc researcher in the Department of Computer Science and Technology at Tsinghua University. His research interests focus on machine learning on graphs, including graph neural network (GNN) and network embedding (a.k.a. network representation learning). He has published over a dozen papers in prestigious conferences and journals, including KDD, AAAI, IJCAI, and TKDE.



Peng Cui is an Associate Professor with tenure in Tsinghua University. He got his PhD degree from Tsinghua University in 2010. His research interests include causally-regularized machine learning, network representation learning, and social dynamics modeling. He has published more than 100 papers in prestigious conferences and journals in data mining and multimedia. His recent research won the IEEE Multimedia Best Department Paper Award, SIGKDD 2016 Best Paper Finalist, ICDM 2015 Best Student Paper Award, SIGKDD 2014 Best Paper Finalist, IEEE ICME 2014 Best Paper Award, ACM MM12 Grand Challenge Multimodal Award, and MMM13 Best Paper Award. He is PC co-chair of CIKM2019 and MMM2020, SPC or area chair of WWW, ACM Multimedia, IJCAI, AAAI, etc., and Associate Editors of IEEE TKDE, IEEE TBD, ACM TIST, and ACM TOMM etc. He received ACM China Rising Star Award in 2015, and CCF-IEEE CS Young Scientist Award in 2018. He is now a Distinguished Member of ACM and CCF, and a Senior Member of IEEE.

Jian Pei is a Professor in the School of Computing Science at Simon Fraser University. He is a renown leading researcher in the general areas of data science, big data, data mining, and database systems. He is recognized as a Fellow of the Royal Society of Canada (Canada's national academy), the Canadian Academy of Engineering, ACM and IEEE. He is one of the most cited authors in data mining, database systems, and information retrieval. Since 2000, he has published one textbook, two monographs and over 300 research papers in refereed journals and conferences, which have been cited extensively by others. His research has generated remarkable impact substantially beyond academia. For example, his algorithms have been adopted by industry in production and popular open source software suites. Jian Pei also demonstrated outstanding professional leadership in many academic organizations and activities. He was the editor-in-chief of the IEEE Transactions of Knowledge and Data Engineering (TKDE) in 2013-16, the chair of ACM SIGKDD in 2017-2021, and a general co-chair or program committee co-chair of many premier conferences. He maintains a wide spectrum of industry relations with both global and local industry partners. He is an active consultant and coach for industry. He received many prestigious awards, including the 2017 ACM SIGKDD Innovation Award, the 2015 ACM SIGKDD Service Award, the 2014 IEEE ICDM Research Contributions Award, the British Columbia Innovation Council 2005 Young Innovator Award, an NSERC 2008 Discovery Accelerator Supplements Award, an IBM Faculty Award (2006), a KDD Best Application Paper Award (2008), an ICDE Influential Paper Award (2018), a PAKDD Best Paper Award (2014), and a PAKDD Most Influential Paper Award (2009).

Xin Wang is currently an Assistant Professor at the Department of Computer Science and Technology, Tsinghua University. He got both of his Ph.D. and B.E degrees in Computer Science and Technology from Zhejiang University, China. He also holds a Ph.D. degree in Computing Science from Simon Fraser University, Canada. His research interests include relational media big data analysis, multimedia intelligence and recommendation in social media. He has published several high-quality research papers in top conferences including ICML, KDD, WWW, SIGIR ACM Multimedia etc. He is the recipient of 2017 China Postdoctoral innovative talents supporting program. He receives the ACM China Rising Star Award in 2020.

Wenwu Zhu is currently a Professor of the Computer Science Department of Tsinghua University. Prior to his current post, he was a Senior Researcher and Research Manager at Microsoft Research Asia. He was the Chief Scientist and Director at Intel Research China from 2004 to 2008. He worked at Bell Labs New Jersey as a Member of Technical Staff during 1996-1999. He received his Ph.D. degree from New York University in 1996.

He served as the Editor-in-Chief for the IEEE Transactions on Multimedia (T-MM) from January 1, 2017, to December 31, 2019. He has been serving as Vice EiC for IEEE Transactions on Circuits and Systems for Video Technology (TCSVT) and the chair of the steering committee for IEEE T-MM since January 1, 2020. His current research interests are in the areas of multimedia computing and networking, and big data. He has published over 350 papers in the referred journals and received nine Best Paper Awards including IEEE TCSVT in 2001 and 2019, and ACM Multimedia 2012. He is an IEEE Fellow, AAAS Fellow, SPIE Fellow and a member of the European Academy of Sciences (Academia Europaea).

APPENDIX A
ADDITIONAL EXPERIMENTAL RESULTS

A.1 Node Classification

The results of node classification on 7 real-world social networks using GAT and GraphSAGE as the base GNN model are reported in Table 7 and Table 8, respectively. In general, the results show similar trends as using GCN (Table 2 in Section 5.2.2). One exception in Table 8 using GraphSAGE as the base GNN is that SAGE_{DW} slightly outperforms $\text{Eigen-SAGE}_{\text{struc}}$ on two datasets (Columbia and Cornell). However, when combining with features, $\text{Eigen-SAGE}_{\text{feat+struc}}$ still outperforms $\text{SAGE}_{\text{feat+DW}}$.

The results of using different features on synthetic datasets with GCN being the backend are shown in Figure 6. Similar to real-world graphs, EigenGCN performs much better than heuristics, such as degree or one-hot, and embedding methods like DeepWalk.

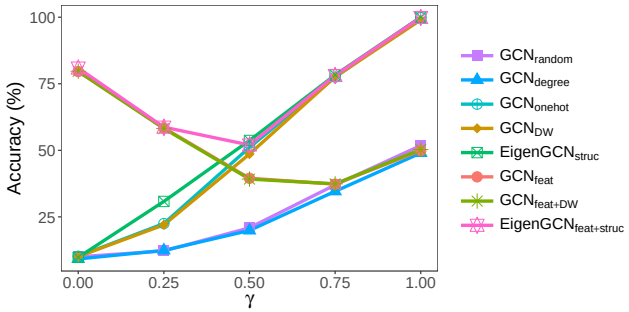


Fig. 6: The experimental results on synthetic datasets when varying γ between 0 and 1. Best viewed in color.

APPENDIX B
ADDITIONAL EXPERIMENTAL DETAILS FOR REPRODUCIBILITY

B.1 Synthetic datasets

As introduced in Section 3.1, we generate synthetic datasets where we control whether the task is feature-driven or structure-driven. Specifically, we use the Stochastic Blockmodel to generate the graph structures, which is a representative method in constructing community graphs. Specifically, we generate $5,000$ nodes in a graph, which are randomly assigned to 1 of the 10 different communities. Within each community, nodes have a probability $prob_{in} = 0.025$ to form edges, while nodes in different communities have a probability $prob_{out} = 0.001$ to form edges.

For node features, we also divide nodes into 10 groups. For each group, we generate a group vector following $\mathcal{N}(0, 4\mathbf{I})$ with dimensionality 32. We constrain that the minimum Euclidean distance between every two group vectors is larger than 1. Node features are randomly generated following i.i.d standard normal distribution $\mathcal{N}(0, \mathbf{I})$ around the group vectors of the groups that the nodes belong to. In this way, nodes in the same group share similar features.

In summary, each synthetic dataset has $5,000$ nodes, $85,294$ edges, 32 features, and 10 labels/classes. Here, a class contains all nodes carrying the same label. The differences among different datasets are the nodes carrying

TABLE 7: The accuracy (%) of node classification using GAT as the base model. The best results with and without node features, respectively, are in bold. A, X, Y stands for graph structures, node features, and node labels, respectively.

Data	Method	Harvard	Columbia	Stanford	Yale	Cornell	Dartmouth	UPenn
A,Y	$\text{GAT}_{\text{random}}$	$\mathbf{81.5 \pm 0.7}$	74.4 ± 1.2	74.8 ± 1.6	80.7 ± 1.8	67.4 ± 2.5	79.6 ± 1.2	74.8 ± 1.5
	$\text{GAT}_{\text{degree}}$	67.5 ± 6.1	63.7 ± 4.1	63.2 ± 2.8	70.5 ± 3.5	54.2 ± 2.6	68.5 ± 3.2	60.9 ± 3.3
	GAT_{DW}	$\mathbf{81.5 \pm 1.3}$	75.4 ± 1.7	76.3 ± 2.2	81.2 ± 2.3	70.1 ± 2.9	77.4 ± 2.2	75.1 ± 1.9
	Eigen-$\text{GAT}_{\text{struc}}$	$\mathbf{81.5 \pm 1.8}$	$\mathbf{77.1 \pm 1.4}$	$\mathbf{78.1 \pm 1.5}$	$\mathbf{83.9 \pm 1.3}$	$\mathbf{72.0 \pm 2.0}$	$\mathbf{81.5 \pm 2.0}$	$\mathbf{78.5 \pm 1.1}$
A,X,Y	GAT_{feat}	73.7 ± 1.6	73.9 ± 2.2	71.5 ± 2.0	72.7 ± 3.5	63.9 ± 2.2	74.9 ± 2.1	69.0 ± 3.5
	$\text{GAT}_{\text{feat+DW}}$	84.3 ± 0.6	78.4 ± 1.7	76.0 ± 3.4	80.9 ± 2.0	73.3 ± 2.6	78.1 ± 1.7	75.5 ± 2.2
	Eigen-$\text{GAT}_{\text{feat+struc}}$	$\mathbf{85.5 \pm 1.3}$	$\mathbf{78.9 \pm 1.1}$	$\mathbf{79.7 \pm 1.4}$	$\mathbf{84.3 \pm 1.8}$	$\mathbf{73.6 \pm 1.8}$	$\mathbf{82.7 \pm 1.2}$	$\mathbf{80.0 \pm 1.7}$

TABLE 8: The accuracy (%) of node classification using GraphSAGE as the base model. The best results with and without node features, respectively, are in bold. A, X, Y stands for graph structures, node features, and node labels, respectively.

Data	Method	Harvard	Columbia	Stanford	Yale	Cornell	Dartmouth	UPenn
A,Y	$\text{SAGE}_{\text{random}}$	14.2 ± 1.2	18.8 ± 1.4	17.8 ± 0.7	17.0 ± 1.9	18.3 ± 1.1	17.8 ± 1.0	18.4 ± 1.2
	$\text{SAGE}_{\text{degree}}$	25.9 ± 2.3	25.9 ± 2.3	27.1 ± 2.0	30.0 ± 3.0	34.0 ± 1.9	25.7 ± 2.3	24.1 ± 1.7
	SAGE_{DW}	82.6 ± 1.0	$\mathbf{77.5 \pm 1.3}$	76.0 ± 1.5	82.2 ± 1.1	$\mathbf{71.5 \pm 1.8}$	80.5 ± 1.5	77.6 ± 1.1
	Eigen-$\text{SAGE}_{\text{struc}}$	$\mathbf{83.1 \pm 0.9}$	76.4 ± 1.6	$\mathbf{78.2 \pm 1.0}$	$\mathbf{84.9 \pm 1.3}$	70.7 ± 1.4	$\mathbf{82.4 \pm 1.6}$	$\mathbf{78.3 \pm 1.5}$
A,X,Y	$\text{SAGE}_{\text{feat}}$	76.6 ± 1.5	77.6 ± 1.5	73.3 ± 1.5	79.2 ± 2.5	62.7 ± 1.9	78.5 ± 1.1	71.4 ± 1.7
	$\text{SAGE}_{\text{feat+DW}}$	80.4 ± 0.9	77.5 ± 2.0	75.1 ± 1.6	82.6 ± 1.1	71.2 ± 1.7	77.3 ± 1.3	77.4 ± 1.1
	Eigen-$\text{SAGE}_{\text{feat+struc}}$	$\mathbf{84.8 \pm 1.2}$	$\mathbf{80.0 \pm 1.3}$	$\mathbf{78.0 \pm 0.9}$	$\mathbf{86.2 \pm 1.4}$	$\mathbf{73.1 \pm 2.0}$	$\mathbf{84.6 \pm 1.4}$	$\mathbf{79.5 \pm 1.7}$

different labels, which corresponds to whether the task is feature-driven or structure-driven, i.e., if the final node labels depend more on graph structures (larger γ), the task is more structure-driven, and if the final node labels depend more on node features (small γ), the task is more feature-driven. For dataset splits, in Figure 1a, the number of randomly selected nodes per class used for training is used as the x-axis and we randomly selected another 200 nodes per class for validation. The rest nodes are used for testing. In Figure 1b, we use 50 nodes per class for training, 150 nodes per class for validation, and the rest for testing.

B.2 Source codes adopted

We use the following publicly available implementations of base GNN models and DeepWalk:

- GCN [9]: https://github.com/tkipf/gcn with MIT License
- StochasticGCN [73]: https://github.com/thu-ml/stochastic_gcn with MIT License
- GIN [25]: https://github.com/PurdueMINDS/RelationalPooling with license unspecified
- SEAL [66]: https://github.com/muhanzhang/SEAL with license unspecified
- DeepWalk [41]: https://github.com/phanein/deepwalk with GPL-3.0 License
- GAT [24] and GraphSAGE [22]: https://github.com/rusty1s/pytorch_geometric with MIT License

B.3 Hyper-parameters

We adopt the following hyper-parameters.

- Synthetic: the number of GCN layers and fully connected layers are given in Section 3.1, with each layer containing 16 units. The dropout rate is searched from $\{0, 0.5\}$ and L2 regularization is 5×10^{-4}. The learning rate is searched from $\{0.005, 0.01, 0.025\}$. The maximum number of epochs is 400 with an early stopping round 100. We also test whether adding residual connections can improve the performance of GCNs and add residual connections in GCN2, GCN3, and GCN5. The hyper-parameters for DeepWalk are the default settings in the implementations of the authors, i.e., the number of walks 10, the walk length 40, the window size 5, and the dimensionality is set to 32 to fairly compare with GCNs with features as inputs. The dimensionality of the eigenspace is also set to $d = 32$ and we set $f(x) = x$ in Eq.(4).
- Harvard, Columbia, Stanford, Yale, Cornell, Dartmouth, and UPenn: we set the same hyper-parameters for GCN, GAT, and GraphSAGE, which is a two-layer architecture with the hidden layer containing 64 units (for GAT, 8 heads with each head containing 8 units), the learning rate is 0.01 with weight decay 5×10^{-4}, the maximum number of epochs is 400, and we select the epoch with the highest validation accuracy. We tried increasing model sizes such as adding more layers and more hidden units, but did not observe consistent improvements. The dropout rate is searched from $\{0, 0.5\}$ and the dimensionality of the eigenspace is searched from $\{128, 256\}$. We set $f(x)$ as a normalization function according to the Frobenius norm:

$$f(\mathbf{Z}) = \frac{\mathbf{Z}}{\|\mathbf{Z}\|_F}. \tag{9}$$

- Cora, Citeseer, and Pubmed: we follow the literature and use the same GCN structure and hyper-parameters as in the original paper [9], i.e., a two-layer GCN with the hidden layer containing 16 units, the dropout rate is 0.5, L2 regularization is 5×10^{-4}, and the learning rate is 0.01. The maximum number of epochs is 400 with

an early stopping round 100. The dimensionality of the eigenspace is $d = 32$ and $f(x)$ is the same as Eq. (9).

- Reddit: since the dataset has more than 200 thousand nodes and 11 million edges, a full-batch training of GNN is infeasible. Instead, we adopt the neighborhood sampling strategy proposed in Stochastic GCN [73] for all the methods. We use the exact same hyper-parameters and sampling strategy suggested in [73], i.e., the GraphSAGE mean pooling architecture, two graph convolution layers with the hidden layer containing 128 units, sampling two neighbors per node, no weight decay, the dropout rate is 0.2, with layer-normalization, the batch-size is 512, and a maximum of 30 epochs. The dimensionality of the eigenspace is set to $d = 128$ and we set $f(x)$ as Eq. (9). We adopt the transductive setting, i.e., training on the whole graph structure.
- CSL: we tested using the default GIN structure (5 GNN layers, 2 MLP layers) [25] and a simplified GIN structure (1 GNN layer, 1 MLP layer). Since no substantial difference is observed, we adopt the latter due to its simplicity. Other settings are the same as in [48], i.e., no dropout or batch normalization, all layers having 16 hidden units, training epsilon via back-propagation, the learning rate 0.01, a maximum of 200 epochs, and no early stopping. The dimensionality of the eigenspace is $d = 16$ and we set $f(x) = |x|$ to ensure permutation-equivariance.
- The eight link prediction datasets: we use the default SEAL architecture [73], i.e., 4 graph convolution layers as in [28] with 32, 32, 32, and 1 units, 1 SortPooling layer with a threshold such that 60% graphs have nodes less than the threshold, 2 1-D convolution layers (with 16 and 32 channels), and 1 dense layer (with 128 units). The learning rate is 10^{-4}, the batch size is 50, the number of epochs is 50 with an early stopping round 20, and the hop number is 1. The dimensionality of the eigenspace is $d = 128$ and we set $f(x)$ as Eq. (9). We additionally search $\mathcal{G}(\mathbf{A})$ from $\left\{\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}, \tilde{\mathbf{A}}\right\}$.

B.4 Hardware and Software Configurations

All experiments are conducted on a server with the following configurations.

- Operating System: Ubuntu 18.04.1 LTS
- CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz
- GPU: GeForce GTX TITAN X with 12GB of Memory
- Software: Python 3.6.9, TensorFlow 1.14.0, PyTorch 1.4.0, PyTorch Geometric 1.4.3, SciPy 1.4.1, NumPy 1.18.2, Cuda 10.1

APPENDIX C
PROOFS

C.1 Proof of Theorem 1

Proof. Denote by $\mathbf{A}$ and $\mathbf{A}'$, respectively, the adjacency matrices of $\boldsymbol{G}$ and $\boldsymbol{G}'$. Since $\mathcal{E}(i,j) = \mathcal{E}'(\mathcal{B}(i), \mathcal{B}(j))$, we have $\mathbf{A}_{i,j} = \mathbf{A}'_{\mathcal{B}(i),\mathcal{B}(j)}$, $\mathcal{F}(\mathbf{A})_{i,j} = \mathcal{F}(\mathbf{A}')_{\mathcal{B}(i),\mathcal{B}(j)}$, and $\mathcal{G}(\mathbf{A})_{i,j} = \mathcal{G}(\mathbf{A}')_{\mathcal{B}(i),\mathcal{B}(j)}$, where $\mathcal{F}(\mathbf{A})$ and $\mathcal{G}(\mathbf{A})$ is the graph structure function in GNNs and the graph structure function for the eigenspace, defined in Eq. (1) and Eq. (4), respectively.

First, let's assume that the l^{th} hidden layer in Eigen-GNN is permutation-equivariant, i.e., $\mathbf{H}^{(l)}_{i,:} = \mathbf{U}^{(l)}_{\mathcal{B}(i),:}, \forall 1 \leq i \leq N$. Since the message-passing framework maintains permutation-equivariance, the $(l+1)^{th}$ layer is also permutation-equivariant. Specifically, using Eq. (1):

$$\begin{aligned}\mathbf{H}^{(l+1)}_{i,:} &= \sigma(\textstyle\sum_j \mathcal{F}(\mathbf{A})_{i,j}\mathbf{H}^{(l)}_{j,:}\mathbf{W}^{(l)}) = \sigma(\textstyle\sum_j \mathcal{F}(\mathbf{A})_{i,j}\mathbf{U}^{(l)}_{\mathcal{B}(j),:}\mathbf{W}^{(l)}) \\ &= \sigma(\textstyle\sum_j \mathcal{F}(\mathbf{A}')_{\mathcal{B}(i),\mathcal{B}(j)}\mathbf{U}^{(l)}_{\mathcal{B}(j),:}\mathbf{W}^{(l)}) = \mathbf{U}^{(l+1)}_{\mathcal{B}(i),:}.\end{aligned} \tag{10}$$

Next, by induction, we only need to prove that the 0^{th} layer, i.e., $\mathbf{H}^{(0)}$ and $\mathbf{U}^{(0)}$, is also permutation-equivariant. The node features are already permutation-equivariant by assumption. For the eigenspace, since the top-d eigenvalues of $\mathcal{G}(\mathbf{A})$ are unique, from linear algebra knowledge, we know that the eigenvectors are determined up to a sign for isomorphism graphs [72]. Since we have adopted $f(x) = |x|$, we have $\mathbf{H}^{(0)}_{i,j} = \mathbf{U}^{(0)}_{\mathcal{B}(i),j}$. The theorem follows. □

C.2 Proof of Theorem 2

Proof. Denote by $\mathbf{H}^{(l)}$ and $\mathbf{U}^{(l)}$, respectively, the representations of the nodes in the l^{th} hidden layer in Eigen-GNN and SGC. Since the last layer in both models os task-specific, e.g., a softmax layer for node classification tasks or a readout function for graph classification tasks, we only need to prove that $\mathbf{U}^{\inf}$ converges to $\mathbf{H}^{(0)} = \mathbf{q}$, where $\mathbf{q} = \mathbf{Q}_{:,1}$ is the one-dimensional eigenspace.

The hidden representations of SGC are calculated as follows [11]:

$$\mathbf{U}^{(l)} = \left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)^l \mathbf{X}. \tag{11}$$

From linear algebra, the asymptotic behavior of $\mathbf{U}^{\inf}$ depends on the spectrum of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$. Denote by $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N$ the eigenvalues of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$. From the spectral graph theory [50], we have the following results.

- The spectrum is between $[-1, 1]$, i.e., $-1 \leq \lambda_1 \leq \lambda_N \leq 1$.
- If the graph is connected, $\lambda_N = 1$ and $\lambda_{N-1} < 1$.
- $\lambda_1 = -1$ if and only if the graph is bipartite.

Combining the above results, we have the following equation for a connected non-bipartite graph.

$$-1 < \lambda_1 \leq ... \leq \lambda_{N-1} < \lambda_N = 1. \tag{12}$$

Then, we know that $\mathbf{U}^{\inf}$ converges to the eigenvector corresponding to λ_N, which is also the eigenvector corresponding to the largest absolute eigenvalue of $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, i.e.,

$$\lim_{l\to\inf} \mathbf{U}^{(l)}_{:,i} = \mathbf{q}, \forall i. \tag{13}$$

□