OPEN: Orthogonal Propagation with Ego-Network Modeling

Liang Yang [1,*], **Lina Kang** [1,*], **Qiuliang Zhang** [1,*], **Mengzhe Li** [1], **Bingxin Niu** [1], **Dongxiao He** [2], **Yuanfang Guo** [3], **Zhen Wang** [4,5], **Chuan Wang** [6], **Xiaochun Cao** [7]
[1]School of Artificial Intelligence, Hebei University of Technology, Tianjin, China
[2]College of Intelligence and Computing, Tianjin University, Tianjin, China
[3]School of Computer Science and Engineering, Beihang University, Beijing, China
[4] OPtics and ElectroNics (iOPEN), Northwestern Polytechnical University, Xi'an, China
[5]School of Cybersecurity, Northwestern Polytechnical University, Xi'an, China
[6]State Key Laboratory of Information Security, IIE, CAS, Beijing, China
[7]School of Cyber Science and Technology, Sun Yat-sen University, Shenzhen, China
email

Abstract

To alleviate the unfavorable effect of noisy topology in Graph Neural networks (GNNs), some efforts perform the local topology refinement through the pairwise propagation weight learning and the multi-channel extension. Unfortunately, most of them suffer a common and fatal drawback: irrelevant propagation to one node and in multi-channels. These two kinds of irrelevances make propagation weights in multi-channels free to be determined by the labeled data, and thus the GNNs are exposed to overfitting. To tackle this issue, a novel Orthogonal Propagation with Ego-Network modeling (OPEN) is proposed by modeling relevances between propagations. Specifically, the relevance between propagations to one node is modeled by whole ego-network modeling, while the relevance between propagations in multi-channels is modeled via diversity requirement. By interpreting the propagations to one node from the perspective of dimension reduction, propagation weights are inferred from principal components of the ego-network, which are orthogonal to each other. Theoretical analysis and experimental evaluations reveal four attractive characteristics of OPEN as modeling high-order relationships beyond pairwise one, preventing overfitting, robustness, and high efficiency.

1 Introduction

Graph Neural Networks (GNNs) have been proven to be a powerful tool to explore irregular graph data by seamlessly combining graph topology and node attributes for node representation learning [1, 2]. From the perspective of spectral graph theory, GNNs, such as GCN [3] and ChebyNet [4], are proposed from graph signal filtering, whose filters are derived from the topology of whole graph, and their success is attributed to low-passing filtering. From the perspective of spatial propagation, GNNs, such as GraphSage [5] and MPNN [6], are presented by following the aggregation and combination scheme for node attributes smoothing. Recent progress demonstrates the equivalence between these two perspectives [7]. However, both the filters based on graph topology and node attribute propagation between neighbourhoods take the implicit assumption that the graph topology is perfect and creditable [8]. Consequently, the oversmoothing issue [9, 10] and the expressive power loss [11], which are caused by overusing topology via stacking multiple layers, are identified.

[*]Equal contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Actually, the graph topology exists large amount of noises. Thus, graph topology refinement is critical to the accuracy and robustness of GNNs [12, 13]. The local refinement, especially through pairwise propagation weight learning, is widely investigated and employed due to the high accuracy and low computation load. As a representative local refinement, Graph Attention Network (GAT) [14] formulates the propagation weight on each edge as the attention between the two connected nodes, and introduces multi-channel propagation via multi-head attention to stabilize the learning process. Afterward, a number of approaches are proposed by following the scheme of propagation weight learning and multi-channel extension. For example, GaAN [15], PGCN [16] and DMP [17] extend GAT via different pairwise propagation weight learning functions. FAGCN [18] relaxes the non-negative constraint in GAT to a real number. ADSF [19] extends GAT to incorporate topology structure into the attention mechanism. GATv2 [20] removes the ranking irrelevance limit in GAT.

Unfortunately, most GNNs, especially those with local refinement, suffer a common but fatal drawback: **irrelevant propagations**. Firstly, the propagations to each node are irrelevant, since propagation weights are either predefined according to the topology or learned based on the contents of the two connected nodes. Actually, the propagations to one node should reflect global characteristics of its ego-network. Secondly, propagations in multi-channels are irrelevant, since parameters to channels are free to be learned without any specific constraints. Actually, to obtain stable and complementary representations from different channels, propagation schemes in different channels should be diverse [17]. These two kinds of irrelevances make propagation weights free to be determined by the labelled data for the specific task, thus resulting GNNs being susceptible to overfitting [21].

To tackle this issue, a novel Orthogonal Propagation with Ego-Network modeling (OPEN) is proposed by modeling these two kinds of relationships between propagations. Firstly, the relevance between propagations to one node is modeled by whole ego-network modeling. Specifically, by interpreting the propagations to one node from the perspective of dimension reduction, propagation weights are inferred from the principal component of ego-network, which corresponds to the mapping function maximizing the variance and captures the characteristics of the whole ego-network. Principal component can be obtained via the eigenvalue decomposition of the covariance matrix, which is efficient for the ego-network. Secondly, the propagations in multi-channels are implemented by employing *all* principal components, each of which corresponds to one channel. Since principal components are orthogonal to each other, the propagations in multi-channel are diverse. The proposed OPEN possesses some attractive characteristics: modeling high-order relationships beyond pairwise one, robustness, and high efficiency. Finally, theoretical analysis demonstrates that the two components of OPEN, i.e., Ego-Network modeling and Orthogonal Propagation in multi-channel, can prevent the over-smoothing issue.

The main contributions of this paper are summarized as follows:

- We investigate the common but fatal issue in GNNs, i.e., irrelevant propagation.
- We propose a novel Orthogonal Propagation with Ego-Network modeling (OPEN) by interpreting propagations in the ego-network from the perspective of dimension reduction.
- We provide a theoretical analysis of OPEN's capability on preventing oversmoothing issue.
- We experimentally verify the effectiveness and robustness of the proposed OPEN.

2 Preliminaries and Analysis

Notations: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with node set $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ and edge set $\mathcal{E}$, where N is the number of nodes. The topology of graph $\mathcal{G}$ can be represented by its adjacency matrix $\mathbf{A} = [a_{ij}] \in \{0, 1\}^{N \times N}$, where $a_{ij} = 1$ if and only if there exists an edge $e_{ij} = (v_i, v_j)$ between nodes v_i and v_j. The degree matrix $\mathbf{D}$ is a diagonal matrix with diagonal element $d_i = \sum_{i=1}^{N} a_{ij}$ as the degree of node v_i. $\mathcal{N}(v_i) = \{v_j | (v_i, v_j) \in \mathcal{E}\}$ stands for the neighbourhoods of node v_i. $\mathbf{X} \in \mathbf{R}^{N \times F}$ and $\mathbf{H} \in \mathbf{R}^{N \times F'}$ denote the collection of node attribute and representation with the i^{th} row, i.e., $\mathbf{x}_i \in \mathbb{R}^F$ and $\mathbf{h}_i \in \mathbb{R}^{F'}$, corresponding to node v_i, where F and F' stand for the dimensions of attribute and representation.

Preliminaries: Most Graph Neural Networks (GNNs) follow an aggregation-combination strategy [6], where each node representation is iteratively updated by aggregating node representations of

neighbourhoods and combining the aggregated representation with the node representation itself as follows

$$\bar{\mathbf{h}}_v^k = \mathrm{AGGREGATE}^k\left(\left\{\mathbf{h}_u^{k-1} | u \in \mathcal{N}(v)\right\}\right), \quad \mathbf{h}_v^k = \mathrm{COMBINATE}^k\left(\mathbf{h}_v^{k-1}, \bar{\mathbf{h}}_v^k\right), \tag{1}$$

where $\bar{\mathbf{h}}_v^k$ stands for the aggregated representation from neighbourhoods. The aggregation operation is the most critical part of the message passing framework, and most GNNs utilize summarization or average function as the implementation of $\mathrm{AGGREGATE}^k$. Therefore, most GNNs can be unified under the following formula

$$\mathbf{h}_v^k = \sigma\Bigg(\sum_{u \in \mathcal{N}(v)} c_{uv}^k \mathbf{h}_u^{k-1} \mathbf{W}^k\Bigg), \tag{2}$$

where $\mathbf{W}^k$ is the learnable parameter and the $\sigma(\cdot)$ is the nonlinear mapping function. The scalar c_{uv} is the averaging weight, which determines the scheme of aggregation. GNNs can be divided into two categories according to the design of c_{uv}. The methods in the first category fix c_{uv} by regarding topology information as perfect.For example, GCN [3] and SGC [22] set $c_{uv}^k = 1/(\sqrt{(d_u+1)(d_v+1)}$, while GIN [23] sets $c_{uv}^k = 1$ for $u \neq v$ and $c_{vv}^k = 1 + \epsilon^k$. The methods in the second category tend to learn c_{uv} by considering topology as noisy. For example, GPRGNN [24] sets $c_{uv}^k = \gamma^k/(\sqrt{(d_u+1)(d_v+1)}$ with γ^k as learnable real number. Graph Attention Network (GAT) [14], Gated Attention Network (GaAN) [15] and Probabilistic GCN [16] model the propagation weights as the function of the attributes of connecting nodes via normalized attention mechanism as

$$c_{uv} = softmax(e_{uv}) = \exp(e_{uv}) / \sum_{k \in \mathcal{N}(u)} \exp(e_{uk}). \tag{3}$$

where e_{uv} denotes the similarity between nodes v and u. The similarity function of attributes can be specified as $e_{uv}^{GAT} = \mathrm{LeakyReLU}(\mathbf{b}^k[\mathbf{W}\mathbf{h}_u^{k-1} || \mathbf{W}\mathbf{h}_v^{k-1}])$, $e_{uv}^{GaAN} = (\mathbf{W}\mathbf{h}_u^{k-1})^T \mathbf{O}^k (\mathbf{W}\mathbf{h}_v^{k-1})$ and $e_{uv}^{PGCN} = -(\mathbf{W}\mathbf{h}_u^{k-1} - \mathbf{W}\mathbf{h}_v^{k-1})^T \mathbf{\Sigma} (\mathbf{W}\mathbf{h}_u^{k-1} - \mathbf{W}\mathbf{h}_v^{k-1})$ where $\mathbf{b}^k$, $\mathbf{O}^k$ and $\mathbf{\Sigma}^k$ are learnable parameters. Some efforts have been paid to simplify them, such as setting $\mathbf{O}^k$ as identity matrix, i.e., $\mathbf{O}^k = \mathbf{I}$ or constraining $\mathbf{\Sigma}^k$ as diagonal matrix. FAGCN [18] relaxes the non-negative constraint in GAT to real number by employing $tanh(\cdot)$ instead of softmax.

GAT [14] proposes to improve the stability by employing multi-head attentions as in Transformer [25]. The multi-head attention essentially performs multi-channel propagation. DMP [17] formates multi-channel propagation as diverse message passing by enhancing Eq. (2) via feature-wise propagation weights as

$$\mathbf{h}_v^k = \sigma\Bigg(\sum_{u \in \mathcal{N}(v)} \mathbf{c}_{uv}^k \odot \mathbf{h}_u^{k-1} \mathbf{W}^k\Bigg), \tag{4}$$

where $\odot$ denotes the element-wise product of vectors and the learnable propagation weight vector $\mathbf{c}_{vu}^k$ has the same length as the node representation $\mathbf{h}_u^{k-1}$. To reduce the model complexity, DMP presents two schemes to efficiently learn $\mathbf{c}_{vu}^k$'s.

Analysis: By analyzing the single-channel propagation in Eq. (4) and the multi-channel propagation in Eq. (4), it can be observed that existing methods possess two serious drawbacks as shown in the blue dashed box in Fig. 1:

- **Propagations to each node are irrelevant.** As shown in Eq. (3), the propagation weight, which is based on the contents of the two connected nodes, models *pairwise* relationship. Thus, the different propagations to each node are irrelevant to each other. However, the node representation, which is obtained from aggregation over neighbourhood, should reflect global characteristics of its neighbourhood, such as high-order relationship.
- **Propagations in multi-channels are irrelevant.** As shown in Eq. (4), propagations in different channels are free to learn without any specific constraints. Thus, the propagation weights in different channels may be similar and redundant. However, to obtain stable and complementary representations from different channels, propagation schemes in different channels should be diverse enough.

In summary, neither the relationship between the propagations to each node nor the relationship between propagations in different channels is considered. Thus, propagation weights are free to be determined by the labelled and the specific task, and thus the models tend to be overfitting [21].

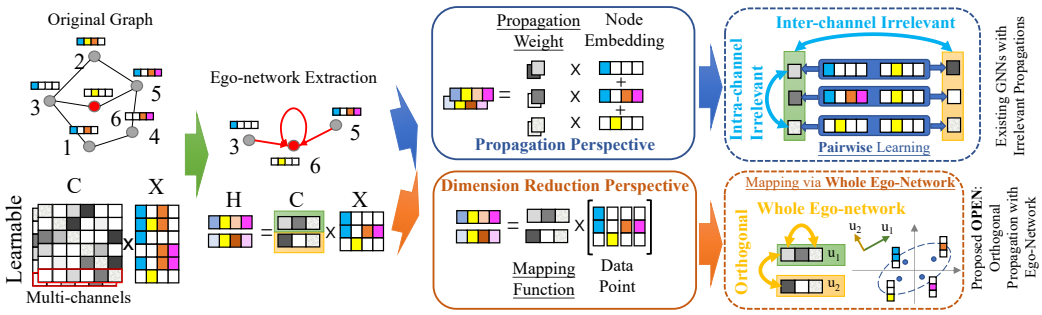

Figure 1: Comparisons between existing GNNs with irrelevant propagations and the proposed OPEN (Orthogonal Propagation with Ego-Network modeling). The essential difference between them is the different perspectives on ego-network modeling. **Upper Part**: Existing GNNs are from the perspective of propagation with pairwise propagation weights learning, and thus both inter-channel and intra-channel propagations are irrelevant. **Lower Part:** The proposed OPEN is from the perspective of dimension reduction of data points, and thus mapping functions are from whole ego-network modeling and orthogonal between different channels.

3 Methodology

To tackle the issues mentioned above, Orthogonal Propagation with Ego-Network modeling (OPEN) is proposed by modeling these two kinds of relationships between propagations in subsections 3.1 and 3.2, respectively. Section 3.3 provides implementation details. Finally, section 3.4 theoretically analyzes its capability on preventing over-smoothing issue.

3.1 Ego-Network Modeling

Firstly, the relationship between propagations to one node is modeled. By ignoring the layer k and considering the first layer, the propagation component in Eq. (2) can be reformulated as

$$\mathbf{h}_v = \sum_{u \in \mathcal{N}(v)} c_{uv} \mathbf{h}_u = \mathbf{c}_v \mathbf{H}_v, \tag{5}$$

where $\mathbf{H}_v \in \mathbb{R}^{|\mathcal{N}_v| \times F}$ stands for the matrix containing the representations of nodes in $\mathcal{N}_v$ and $\mathbf{c}_v \in \mathbb{R}^{|\mathcal{N}_v|}$ represents the propagation weights from nodes in $\mathcal{N}_v$ to node v. Eq. (5) can be interpreted from two perspectives as shown in Fig. 1. From the perspective of message passing, the representation of node v, i.e., $\mathbf{h}_v$ is the aggregation of embeddings of its neighbourhood, which are the rows of $\mathbf{H}_v$, and propagation weights often model pairwise relationship as shown in the blue solid box in Fig. 1. This perspective induces the propagations to each node irrelevant as shown in the blue dashed box in Fig. 1.

From another perspective, $\mathbf{h}_v \in \mathbb{R}^{1 \times F}$ can be regarded as the one-dimensional representations of F data points, each of which corresponds to a $\mathcal{N}_v$-dimensional column vector of $\mathbf{H}_v$. And, $\mathbf{c}_v \in \mathbb{R}^{|\mathcal{N}_v|}$ can be seen as the mapping function, which reduces the dimension from $\mathbb{R}^{|\mathcal{N}_v|}$ to $\mathbb{R}$ as shown in the orange solid box in Fig. 1. From the perspective of dimension reduction, the F representations in $\mathbf{h}_v$ should preserve the discriminative information in original $|\mathcal{N}_v|$-dimensional space of $\mathbf{H}_v$. Therefore, learning the propagation weights $\mathbf{c}_v$ in ego-network of node v is converted to the problem of seeking dimension reduction mapping function.

Among the existing dimension reduction methods, principal component analysis (PCA) is the widely-adopted unsupervised one. The principal components correspond to the mapping functions. To facilitate the description, the subscripts of $\mathbf{H}_v$ and $\mathbf{c}_v$ are removed, and $\mathbf{H}$ is represented as the collection of column vectors as $\mathbf{H} = \{\mathbf{h}_{\cdot,1}, \mathbf{h}_{\cdot,2}, ..., \mathbf{h}_{\cdot,F}\}$, each of which is an $|\mathcal{N}_v|$-dimensional vector. By denoting the mean and covariance matrix of data matrix $\mathbf{H}$ as $\bar{\mathbf{h}} = \frac{1}{F}\sum_{j=1}^{F} \mathbf{h}_{\cdot,j}$ and $\mathbf{S} = \frac{1}{F}\sum_{j=1}^{F}(\mathbf{h}_{\cdot,j} - \bar{\mathbf{h}})(\mathbf{h}_{\cdot,j} - \bar{\mathbf{h}})^T$, respectively, the principal components are the eigenvectors of covariance matrix $\mathbf{S}$, i.e.

$$\mathbf{S}\mathbf{u}_j = \lambda_j \mathbf{u}_j, \quad j = 1, 2, ..., |\mathcal{N}_v| \tag{6}$$

where $\mathbf{u}_j$ is the eigenvector corresponding to the eigenvalue λ_j as shown in the orange dashed box in Fig. 1. By sorting λ_j in descending order, the eigenvector $\mathbf{u}_1$ corresponding to largest eigenvalue λ_1 is the first principal component. By using the $\mathbf{u}_1$ as the mapping function $\mathbf{c}$, the obtained 1-dimensional representation possesses the largest variance, and thus preserves the discriminative ability.

Ego-network of Hub Nodes: The computational complexity, which consists of construction of $\mathbf{S}$ and EVD, is $\mathcal{O}(|\mathcal{N}_v|^2 F)$. Thus, it is efficient for most nodes with small neigbourhood. However, it may be inefficient for nodes with large neighbourhood, such as hub nodes. Thus, we present the treatment for them. By denoting $\tilde{\mathbf{H}} = \{\mathbf{h}_{\cdot,1} - \bar{\mathbf{h}}, \mathbf{h}_{\cdot,2} - \bar{\mathbf{h}}, ..., \mathbf{h}_{\cdot,F} - \bar{\mathbf{h}}\}$, Eq (6) can be written as $\frac{1}{N}\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T\mathbf{u}_j = \lambda_j\mathbf{u}_j$. By multiplying both sides by $\tilde{\mathbf{H}}^T$, it holds

$$\frac{1}{N}\tilde{\mathbf{H}}^T\tilde{\mathbf{H}}\tilde{\mathbf{H}}^T\mathbf{u}_j = \lambda_j\tilde{\mathbf{H}}^T\mathbf{u}_j. \tag{7}$$

By denoting $\mathbf{z}_j = \tilde{\mathbf{H}}^T\mathbf{u}_j$, Eq. (7) and $\mathbf{u}_j$ can be respectively reformulated as

$$\left(\frac{1}{N}\tilde{\mathbf{H}}^T\tilde{\mathbf{H}}\right)\mathbf{z}_j = \lambda_j\mathbf{z}_j, \quad \mathbf{u}_j = \frac{1}{(F\lambda_j)^2}\tilde{\mathbf{H}}\mathbf{z}_j. \tag{8}$$

Since $\frac{1}{N}\tilde{\mathbf{H}}^T\tilde{\mathbf{H}} \in \mathbb{R}^{F\times F}$, its construction and EVD are efficient. Thus, the complexity of ego-network modeling for hub nodes, which possess a large number of neighbourhoods, i.e., $|\mathcal{N}_v|$, is reduced from $\mathcal{O}(|\mathcal{N}_v|^2 F)$ to $\mathcal{O}(|\mathcal{N}_v|F^2)$. Therefore, ego-network for hub node can be efficiently modeled.

This new perspective and corresponding ego-network modeling provide following three attractive characteristics:

- **Beyond pairwise modeling:** Since the covariance matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{N}_v|\times|\mathcal{N}_v|}$ essentially models the correlation between nodes in the ego-network $\mathcal{N}_v$, its eigenvector represents the high-order characteristics of ego-network, which is beyond the pairwise relationship between two connected nodes. Therefore, the propagations to each node are jointly modeled.
- **Preventing overfitting issue:** The mapping function, i.e., the propagation weights, is inferred from the ego-network instead of learning from labelled nodes for specific task. The model parameters of propagation weight learning are omitted, and thus it prevent overfitting.
- **Hight efficiency:** The eigenvalue decomposition (EVD) is performed over the ego-networks whose sizes are much smaller than that of whole network, and thus the EVD is efficient.

3.2 Orthogonal Propagations in Multi-channels

Since multi-channel introduces multiple groups of trainable parameters, it tends to be more serious overfitting than the single-channel case. To tackle this issue and enhance the robustness, an intuitive strategy is to boost the diversities of different channels to provide complementary effects. GNNs may implement this requirement by designing diverse propagation scheme. However, it is difficult to constrain pairwise weight learning to be diverse in an efficient manner, since diversity often requires to impose computationally expensive orthogonality constraints.

Fortunately, the employed PCA in ego-network modeling in section 3.1 simultaneously produces $|\mathcal{N}_v|$ orthogonal components $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{|\mathcal{N}_v|}\}$, which corresponds to different mapping functions. Note that the PCA is efficient on ego-networks with a small number of nodes. Thus, these $|\mathcal{N}_v|$ orthogonal components $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{|\mathcal{N}_v|}\}$ can be employed as the propagation weights in $|\mathcal{N}_v|$ different channels to realize the diversity requirement. By letting $\mathbf{c}_v^j = \mathbf{u}_j$ as the propagation weights over the ego-network of node v on j^{th} channel, Eq. (5) can be extended to multi-channel form as

$$\mathbf{h}_v^j = \mathbf{c}_v^j\mathbf{H}_v, \quad j = 1, 2, ..., |\mathcal{N}_v| \tag{9}$$

Discussion: Ortho-GConv [26] also introduces the orthogonality in GNNs. However, it is very different from the proposed OPEN on both motivation and methodology. From the perspective of motivation, Ortho-GConv tends to alleviate oversmoothing issue, while OPEN aims at overfitting issue. From the perspective of methodology, Ortho-GConv imposes orthogonality on feature transformation $\mathbf{W}^k$, while OPEN presents orthogonal propagation, which is more challenging and essential in GNNs.

3.3 Model Details

Subsections 3.1 and 3.2 provide the basic ideas of ego-network modeling and extension to multi-channels with orthogonal propagations. This section presents the details on implementation.

Firstly, the number of channels is set as J in advance. Since the number of principal components for each ego-network is the same as the number of neighbourhood of the center node, i.e., $|\mathcal{N}|_v$, these numbers varies over nodes. To make representations of nodes comparable, the number of channels is set as a hyper-parameter. For the nodes, whose number of neighboood is less than J, the lacking components are set as zero vector, i.e., $\mathbf{c}_v^j = \mathbf{0}$ for $j = |\mathcal{N}|_v + 1, ..., J$. The tuning experiments on J are shown in subsection 4.5.

Secondly, the final representation of node can be obtained by combing representations in multi-channels from Eq. (9), such as summarization or concatenation. To boost the expressive capability, attention mechanism is employed to combine different representations with shared learnable parameter $\mathbf{b}$, i.e.

$$\mathbf{h}_v = \sum_{j=1}^{J} \alpha_v^j \mathbf{h}_v^j, \quad \alpha_v^j = \frac{\exp(\mathbf{b}^T \mathbf{h}_v^j)}{\sum_{i=1}^{J} \exp(\mathbf{b}^T \mathbf{h}_v^i)}. \tag{10}$$

Similar to existing GNNs, OPEN is trained by feeding the final representation $\mathbf{h}_v$'s into the cross-entropy between predicted and ground-truth labels on labelled nodes for semi-supervised task.

Thirdly, propagation weights inferred from the original node attributes are fixed for all layers. Although it can obtain optimal solution by inferring propagation weights for the next layer from the representations obtained in last layer, the inference process and the back-propagation process can't be seamlessly combined. For the consideration of computational complexity, propagation weights inferred from the original node attributes are employed for all layers.

3.4 Theory Analysis on Preventing Over-smoothing

Over-smoothing issue [9, 10], i.e., representations of all nodes converge to points independent from their original attributes, has been identified as the most serious issue to prevent GNNs from being deep. The over-smoothing also leads to the exponential loss of expressive power for node classification [11]. According to [10], over-smoothing issues is caused by that the eigenvector corresponding to the largest eigenvalue of normalized adjacency matrix ($\lambda_1 = 1$) is determined by degree, i.e., $\mathbf{u}_1 = \mathbf{D}\mathbf{e}$ for asymmetric normalization or $\mathbf{u}_1 = \mathbf{D}^{\frac{1}{2}}\mathbf{e}$ for symmetric normalization, where $\mathbf{e}$ is the vector of ones. Thus, the node representations after infinite layers are

$$\mathbf{H}^{\infty} = \lim_{l \to \infty} \tilde{\mathbf{A}}^l \mathbf{X} = \mathbf{u}_1 \mathbf{u}_1^T \mathbf{X} = \mathbf{u}_1 \left(\mathbf{u}_1^T \mathbf{X}\right) = \mathbf{u}_1 \mathbf{w}^T, \tag{11}$$

where $\mathbf{w} = \mathbf{X}^T \mathbf{u}_1$ is the weighted combination of attributes from all nodes, and thus it is shared by all nodes. Therefore, the final representation $\mathbf{H}^{\infty}$ is determined by node degree $\mathbf{D}$. To alleviate this issue, many efforts have been paid. Some of them modify the topology, such as DropEdge [27] and GRAND [28], while others revise the outputs from layers, such as PairNorm [29] and Inflation [30].

Different from existing methods, the proposed OPEN essentially refines the topology $\mathbf{A}$ according to the attributes of nodes in ego-network $\mathbf{X}_v$, i.e., $\hat{\mathbf{A}} = f(\mathbf{A}, \mathbf{X})$. Therefore, it is intuitive that the $\hat{\mathbf{u}}_1$ corresponding $\hat{\mathbf{A}}$ is the composition of $\mathbf{D}$ and $\mathbf{X}$. According to Eq. (11), the final representation is relevant to $\mathbf{X}$, i.e. preventing over-smoothing issue. The following theorem is proved in Appendix.

Theorem 1. *The representation of each node from OPEN is relevant to the principal components of its corresponding ego-network's attribute.*

4 Evaluations

Firstly, this section provides experimental setups, including dataset, baselines and implementation details. Then, the node classification results are analyzed followed by hyper-parameter tuning. Finally, the capabilities on preventing over-smoothing and overfitting are verified.

Table 1: Statistics of datasets used for node-level tasks.

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Amazon-Computers	13,752	245,861	767	10
Amazon-Photo	7,650	119,081	745	8
Coauthor-CS	18,333	81,894	6,805	15
Coauthor-Physics	34,493	247,962	8,415	5

Table 2: Node classification performance in terms of micro-f1 scores.

dataset	Cora	Pubmed	Citeseer	Computer	Photo	CS	Physics
MLP	74.82±2.22	63.76±0.78	74.05±2.10	70.48±0.28	78.69±0.30	88.30±0.70	88.90±1.10
LogReg	70.10±2.30	61.00±2.20	71.10±3.10	76.80±5.70	79.20±6.50	86.40±0.90	86.70±1.50
LP	78.00±0.20	75.30±0.20	69.00±0.50	70.80±0.00	67.80±0.00	74.30±0.00	90.20±0.50
Chebyshev	82.20±0.50	81.80±0.50	73.40±0.30	72.60±0.00	84.30±0.00	91.50±0.00	92.10±0.30
MoNet	82.30±1.30	80.20±2.00	74.60±2.30	88.60±2.20	91.20±1.30	90.80±0.60	92.50±0.90
GCN	85.77±0.25	88.13±0.28	73.68±0.31	86.51±0.54	92.42±0.22	94.55±0.24	95.58±0.20
GAT	87.37±0.30	87.62±0.26	74.32±0.27	86.93±0.39	92.56±0.35	93.98±0.22	95.63±0.18
GraphSAGE	87.77±1.04	88.42±0.50	71.09±1.30	83.11±0.23	90.51±0.25	OOM	95.52±0.54
SGC	86.20±0.12	87.50±0.25	78.10±0.13	80.09±0.16	88.25±0.52	89.62±0.42	90.05±0.50
GCNII	88.49±2.78	89.57±1.56	77.08±1.21	86.13±0.51	90.98±0.93	93.88±0.40	96.02±0.15
APPNP	87.87±0.85	89.40±0.61	76.53±1.33	81.99±0.26	91.11±0.26	94.92±0.20	95.84±0.22
JKNet	88.93±1.35	87.68±0.30	74.37±1.53	77.80±0.97	87.70±0.70	92.32±0.72	95.71±0.21
C-GAT	88.40±0.30	87.60±0.30	79.90±0.30	OOM	OOM	OOM	OOM
GPRGNN	88.37±1.33	89.05±0.52	78.88±1.70	89.43±0.86	94.34±0.35	94.76±0.20	96.39±0.20
DMP	86.68±1.13	89.67±0.58	76.48±1.73	87.62±0.81	94.06±0.45	**95.02±0.29**	94.74±0.20
OPEN	**89.31±0.53**	**90.05±0.23**	**80.42±0.72**	**90.93±0.29**	**95.43±0.27**	94.99±0.13	**96.92±0.05**

4.1 Datasets

To comprehensively evaluate the proposed OPEN, 7 widely used datasets are employed. Statistics of datasets are shown in Table 1. These datasets can be divided into three categories.

- **Citation Networks.** Cora, Citeseer, and Pubmed, which are widely used to verify GNNs, are standard citation network benchmark datasets [31, 32]. In these networks, nodes and edge represent papers and citations between them, respectively. Words in the paper are employed to represent the node feature in bag-of-word form. The academic topic of paper is taken as the label of node.
- **Co-purchase Networks.** Amazon-Computers (Computers) and Amazon-Photo (Photo) are two networks of co-purchase relationships [33]. In these networks nodes represent goods and edges stand for the connected two goods being frequently bought together. Each node owns a bag-of-words feature extracted from product reviews. The categories of the goods are employed as the label of node.
- **Coauthor Networks.** Coauthor-CS (CS) and Coauthor-Physics (Physics) are two co-author networks based on the Microsoft Academic Graph from the KDD Cup 2016 challenge [33]. In these networks nodes represent authors and edges stand for that the connected two authors have co-authored a paper. Each node owns a bag-of-words feature based on paper keywords. The most active research fields of the authors are employed as the label.

4.2 Baselines

To verify the superiority of the proposed OPEN, 15 baseline methods are employed for performance comparison. These methods are divided into two categories. The first category consists of basic methods for graph data, including the multiple layer perception (MLP), Logistic Regression (LogReg) and Label Propagation (LP) [34], Chebyshev [4], Graph Convolutional Network (GCN) [3], Graph Attention Network (GAT) [14], GraphSAGE [5], Simple Graph Convolution (SGC) [22], MoNet [35].

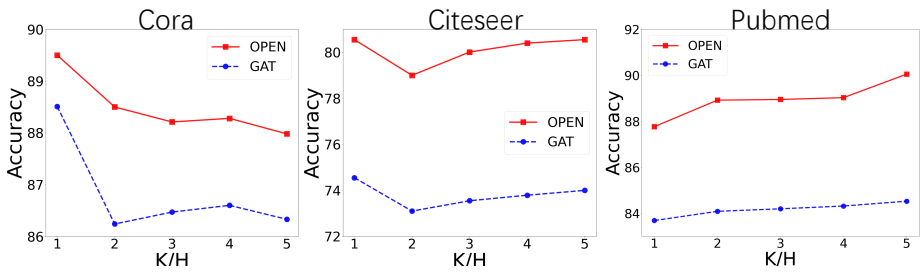

Figure 2: The impact of the number of channels on the performance on Cora, Citeseer and Pubmed.

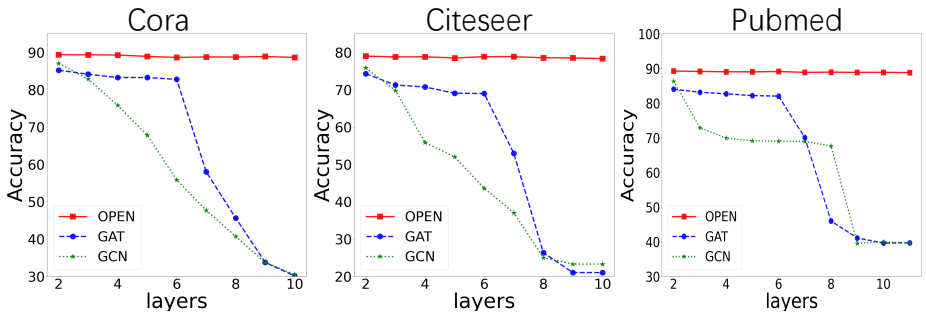

Figure 3: Node classification results with various model depths on Cora, Citeseer and Pubmed.

The methods in the second category possess some attractive characteristics, such as preventing over-smoothing issue and high accuracy, etc This category contains Personalized Propagation of Neural Predictions (APPNP) [36], Jumping Knowledge Networks (JKNet) [37], GCN with Initial residual and Identity mapping (GCNII) [38], Generalized PageRank GNN (GPRGNN) [24], Diverse Message Passing (DMP) [17] and Constrained GAT (C-GAT) [21]. We employ the authors' implementations for all baseline methods with the default hyper-parameters.

4.3 Implementation Details

The proposed OPEN employs 2-layers network with $K = 5$ channels except for the hyper-parameter tuning (Sec. 4.5) and over-smoothing investigation (Sct 4.6). The whole network is trained in an end-to-end manner using the Adam optimizer with an initial learning rate of 0.001. The maximum number of epochs is set up to 1000. Besides, early stopping with a patience of 50 is also utilized. For all datasets, we randomly split nodes of each class in to 60%, 20% and 20% for training validation and testing, and run on test sets over 10 random splits, as suggested in [39]. For fair comparison, the performance of all the methods, including baseline methods, are obtained on the same splits.

4.4 Result Analysis

The node classification results are shown in Tab. 2. The proposed OPEN achieves the new SOTA on 6 networks in all 7 networks. Note that the proposed OPEN not only outperforms all basic GNNs, such as GCN and GraphSAGE, but also significantly beats almost all SOTA methods, such as GCNII, GPRGNN and DMP. These demonstrate the superiority of the proposed OPEN on classification accuracy. This can be attributed to both the effectiveness of ego-network modeling on capturing high-order semantic information and the high diversity representations obtained from orthogonal propagation in multi-channels. Besides, the proposed OPEN also outperforms other extensions to GAT, such as C-GAT and DMP. This indicates that compared to other enhancements to the propagation weights learning and multi-channel learning, ego-network modeling and orthogonal propagation are more effective. In summary, these effectivenesses show that the modeling the irrelevance between propagations is important and necessary for GNNs.

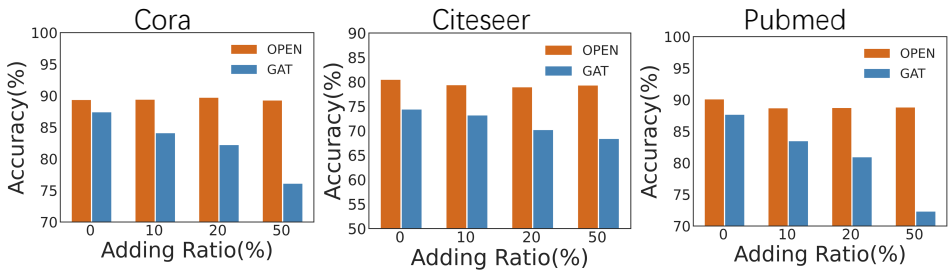

Figure 4: Node classification performance on graphs with randomly adding noisy edges.

4.5 Hyper-parameter Tuning and Ablation Study

The only hyper-parameter in the proposed OPEN is the number of principal components, K, which is also corresponds to the number of channels. This section investigates the impact of K on classification accuracy. To exclude the influences from different datasets, the impact of the number of channels in GAT, i.e., the number of attention heads, is also investigated. Fig.2 shows that OPEN consistently outperforms GAT on all the number of channels. However, the trends with different number of channels are different on different datasets. In most cases, the superiority of OPEN is more remarkable with large K compared to GAT. Thus, the K is set as 5 for other experiments by balancing the accuracy and running time.

Ablation Study: The hyper-parameter tuning experiments also enable the ablation study. 1) When $K = 1$, only the ego-network modeling component works. The significant outperformance compared to GAT demonstrates that the relevant propagation in OPEN is more superior than irrelevant propagations in GAT. 2) The consistent outperformances on all K's reveal the importance of orthogonal propagations in multi-channels, which are also relevant. The ablation study illustrates that both relevance modelings are critical and necessary.

4.6 Preventing Over-smoothing Issue

Section 3.4 theoretically analyzes the capability of OPEN on preventing oversmoothing issue. This section provides experimental evaluations. The node classification performance changes of GCN, GAT and OPEN with various model depths are shown in Fig. 3. GCN tends to be over-smoothing via only a few layers, since it employs topology-induced single-channel propagation. GAT alleviates the overmoothing issue by introducing irrelevant multi-channel propagations. Unfortunately, it also be oversmoothing after few layers. By enhancing the GAT with relevant multi-channel propagations, OPEN significantly prevents the over-smoothing issue. After many layers, the performances of OPEN do not remarkably drop. This matches the theoretical results in Section 3.4, and verifies the capability of relevant propagation on preventing oversmoothing issue.

4.7 Robustness

This section investigates the robustness of the proposed OPEN, which indicates whether the OPEN is able to overcome the overfitting problem, as in [21]. To this end, experiments are conducted by randomly perturbing the edges in the testing graph. Specifically, a set of nodes are randomly selected according to a given sampling ratio, and then one edge is randomly added on these nodes. Fig. 4 reports the classification performance on testing graphs with different ratios of nodes. The performances of GAT significantly drop as the ratios of nodes increase, which indicate it tends to be overfitting to the given data. In contrary, the performance drops of the proposed OPEN are very slight. It indicates the robustness of OPEN. This robustness may be attributed to two aspects. Firstly, the propagation weights of OPEN are inferred from neighbourhoods instead of learning from labelled data. Secondly, the orthogonal propagations in multi-channels promote the diversity, and thus enhance the robustness. This investigation shows that relevant propagation modeling can promote the robustness of the GNNs.

5 Conclusions

This paper identifies the *irrelevant propagations* issue in Graph Neural Networks (GNNs), which makes models exposed to overfitting to the labelled data. The propagation irrelevance includes 1) the propagations to one node are irrelevant, 2) propagations in multi-channels are irrelevant. This paper presents a novel Orthogonal Propagation with Ego-Network modeling (OPEN) to model these two kinds of relevances between propagations. By interpreting the propagations to one node from the perspective of dimension reduction, propagation weights are inferred from the principal components of ego-network, which are orthogonal to each other. Theoretical analysis and experimental evaluations reveal four attractive characteristics of the proposed OPEN as modeling high-order relationship beyond pairwise one, preventing overfitting issue, robustness, and high efficiency. These attractive characteristics reveal the importance of modeling propagation relevance.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China under Grant 61972442, Grant 62102413, Grant U2001202, Grant U1936210, Grant 61876128, in part by the Key Research and Development Project of Hebei Province of China under Grant 20350802D and 20310802D; in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Natural Science Foundation of Tianjin of China under Grant 20JCY-BJC00650, in part by the China Postdoctoral Science Foundation under Grant 2021M703472, and in part by the Fundamental Research Funds for Central Universities.

References

[1] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021.

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

[3] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845, 2016.

[5] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1263–1272, 2017.

[7] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International Conference on Learning Representations*, 2021.

[8] Liang Yang, Zesheng Kang, Xiaochun Cao, Di Jin, Bo Yang, and Yuanfang Guo. Topology optimization based graph convolutional network. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4054–4061, 2019.

[9] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3538–3545, 2018.

[10] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 338–348, 2020.

[11] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[12] Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. Graph structure estimation neural networks. In *WWW*, pages 342–353, 2021.

[13] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *WWW*, pages 1392–1403, 2022.

[14] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[15] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *UAI*, pages 339–349, 2018.

[16] Liang Yang, Yuanfang Guo, Junhua Gu, Di Jin, Bo Yang, and Xiaochun Cao. Probabilistic graph convolutional network via topology-constrained latent space model. *IEEE Trans. Cybern.*, 52(4):2123–2136, 2022.

[17] Liang Yang, Mengzhe Li, Liyang Liu, Bingxin Niu, Chuan Wang, Xiaochun Cao, and Yuanfang Guo. Diverse message passing for attribute with heterophily. In *NeurIPS*, pages 4751–4763, 2021.

[18] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Event, February 2-9, 2021*, pages 3950–3957, 2021.

[19] Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. Adaptive structural fingerprints for graph attention networks. In *ICLR*, 2020.

[20] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *ICLR*, 2022.

[21] Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Improving graph attention networks with large margin-based constraints. *arXiv preprint arXiv:1910.11945*, 2019.

[22] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6861–6871, 2019.

[23] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[24] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[26] Kai Guo, Kaixiong Zhou, Xia Hu, Yu Li, Yi Chang, and Xin Wang. Orthogonal graph neural networks. In *AAAI*, volume abs/2112.14438, 2022.

[27] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[28] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[29] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[30] Dongxiao He, Rui Guo, Xiaobao Wang, Di Jin, Yuxiao Huang, and Wenjun Wang. Inflation improves graph neural networks. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 1466–1474, 2022.

[31] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[32] Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, 2012.

[33] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation, 2019.

[34] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 912–919, 2003.

[35] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5425–5434, 2017.

[36] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[37] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 5449–5458, 2018.

[38] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 1725–1735, 2020.

[39] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Table 3: Node classification results in terms of micro-f1 (20 labelled nodes per class for training).

dataset	Cora	Citeseer	Pubmed	Computer	Photo	CS	Physics
MLP	58.20±2.10	59.10±2.30	70.0±2.10	44.90±5.80	69.60±3.80	88.30±0.70	88.90±1.10
LogReg	57.10±2.30	61.00±2.20	64.10±3.10	64.10±5.70	73.00±6.50	86.40±0.90	86.70±1.50
LP	68.00±0.20	45.30±0.20	63.00±0.50	70.80±0.00	67.80±0.00	74.30±0.00	90.20±0.50
Chebyshev	81.20±0.50	69.80±0.50	74.40±0.30	62.60±0.00	74.30±0.00	91.50±0.00	92.10±0.30
MoNet	81.30±1.30	71.20±2.00	78.60±2.30	83.50±2.20	91.20±1.30	90.80±0.60	92.50±0.90
GCN	81.50±1.30	70.30±0.28	77.80±2.90	76.30±2.40	87.30±1.20	91.10±0.50	92.60±0.70
GAT	81.80±1.30	70.80±0.26	78.50±0.27	78.00±1.90	85.70±1.70	90.50±0.22	91.30±0.60
SGC	81.00±0.00	71.90±0.10	78.90±0.00	74.40±0.01	86.40±0.00	91.00±0.00	90.20±0.40
GCNII	**85.50±0.50**	73.40±0.60	**80.20±0.40**	57.11±13.92	63.03±4.43	88.30±1.25	OOM
APPNP	83.30±0.00	71.80±0.00	80.10±0.00	71.69±4.67	83.62±3.73	91.41±0.44	93.38±0.67
JKNet	81.10±0.00	69.80±0.00	78.10±0.00	64.08±2.10	78.10±7.07	87.07±1.34	92.69±0.73
C-GAT	80.60±0.45	70.99±0.37	79.60±0.11	OOM	OOM	OOM	OOM
GPRGNN	80.55±1.05	68.57±1.22	77.02±2.59	81.71±2.84	91.58±0.87	92.42±0.47	**93.51±0.59**
DMP	80.41±1.48	71.08±1.21	76.29±2.44	71.90±1.84	82.37±1.86	90.44±0.40	90.42±0.55
OPEN	81.68±0.44	**78.37±0.21**	79.35±0.25	**86.01±0.26**	**91.64±0.51**	**94.98±0.11**	92.17±0.56

A Proof of Theorem 1

In this section, the proof to Theorem 1 is given. To this end, some additional notations are defined in advance. Firstly, matrix $\mathbf{I}_v \in \mathbb{R}^{N\times N}$ is defined as the diagonal indicator matrix to indicate the neighbourhood of node v. It i^{th} diagonal element is 1, i.e., $(\mathbf{I}_v)_{ii} = 1$, if and only if node i is the neighbourhood of node v, i.e., $a_{vi} = 1$. Thus, the relationship between the v^{th} row of adjacency matrix $\mathbf{A}$, i.e., $\mathbf{a}_v$, and indicator matrix $\mathbf{I}_v \in \mathbb{R}^{N\times N}$ is

$$(\mathbf{1}\cdot\mathbf{a}_v)\odot\mathbf{I} = \mathbf{I}_v, \tag{12}$$

where $\odot$ denotes the element-wise product, $\mathbf{1}$ denotes vector of ones, and $\mathbf{I}$ stands for identity matrix. Thus, $\hat{\mathbf{H}}_v = \mathbf{I}_v\mathbf{H}$ only remains the rows corresponding to v's neighbourhoods and set other rows as zeros. In other word, $\hat{\mathbf{H}}_v \in \mathbf{R}^{N\times F}$ is the padding version of $\mathbf{H}_v \in \mathbf{R}^{|\mathcal{N}_v|\times F}$ with rows of zeros. Thus, the EVD in Eq. (6) can be extended to

$$\hat{\mathbf{S}}\hat{\mathbf{u}}_j = \lambda_j\hat{\mathbf{u}}_j, \quad j = 1, 2, ..., N \tag{13}$$

where $\hat{\mathbf{S}} = \hat{\mathbf{H}}_v\hat{\mathbf{H}}_v^T \in \mathbb{R}^{N\times N}$ and $\hat{\mathbf{u}}_j \in \mathbb{R}^N$ are the padding versions of $\mathbf{S} \in \mathbb{R}^{|\mathcal{N}_v|\times|\mathcal{N}_v|}$ and $\mathbf{u}_j \in \mathbb{R}^{|\mathcal{N}_v|}$, respectively. Thus, the obtained embedding for node v from our proposed OPEN can be written as

$$\mathbf{h}_v^{OPEN} = \mathbf{u}_1^T\mathbf{H}_v = \hat{\mathbf{u}}_1^T\hat{\mathbf{H}}_v = \hat{\mathbf{u}}_1^T\mathbf{I}_v\mathbf{H} = \left(\hat{\mathbf{u}}_1^T\odot\mathbf{a}_v\right)\mathbf{H}. \tag{14}$$

According to PCA, the mapping $\mathbf{u}_1$ and $\hat{\mathbf{u}}_1$ can be expressed as the combination of the columns of $\mathbf{H}_v$ and $\hat{\mathbf{H}}_v$, i.e.,

$$\mathbf{u}_1 = \sum_{f=1}^{F}\alpha_f\left(\mathbf{H}_v\right)_{\cdot,f} \quad \hat{\mathbf{u}}_1 = \sum_{f=1}^{F}\alpha_f\left(\hat{\mathbf{H}}_v\right)_{\cdot,f}. \tag{15}$$

It indicates that the propagation weights $\mathbf{u}_1$ and $\hat{\mathbf{u}}_1$ are not fully determined by the graph topology, but are mainly impacted by node attribute. Note that different from GAT, where propagation weights are determined by attributes of connected nodes and labels, the propagation weights in OPEN are not impacted by labels any more. Similar to Eq. (14), the embedding obtained from asymmetric adjacency matrix, such as GraphSAGE [5], can be written as

$$\mathbf{h}_v^{topology} = \frac{1}{d_v}\mathbf{1}^T\mathbf{H}_v = \frac{1}{d_v}\mathbf{1}^T\hat{\mathbf{H}}_v = \frac{1}{d_v}\mathbf{1}^T\mathbf{I}_v\mathbf{H} = \left(\frac{1}{d_v}\mathbf{1}^T\odot\mathbf{a}_v\right)\mathbf{H}. \tag{16}$$

Comparing Eqs. (14) and (16), the main difference between OPEN and propagation based on asymmetric adjacency matrix is that the topology-based propagation weight $\frac{1}{d_v}\mathbf{1}$ in propagation based on asymmetric adjacency matrix is replaced by attribute-based propagation weight $\mathbf{u}_1$ (Eq. 15) in the proposed OPEN. Therefore, the representation of node v, i.e., $\mathbf{h}_v^{OPEN}$, is relevant to the principal components of node v's ego-network's attribute, i.e., $\hat{\mathbf{u}}_1^T$.

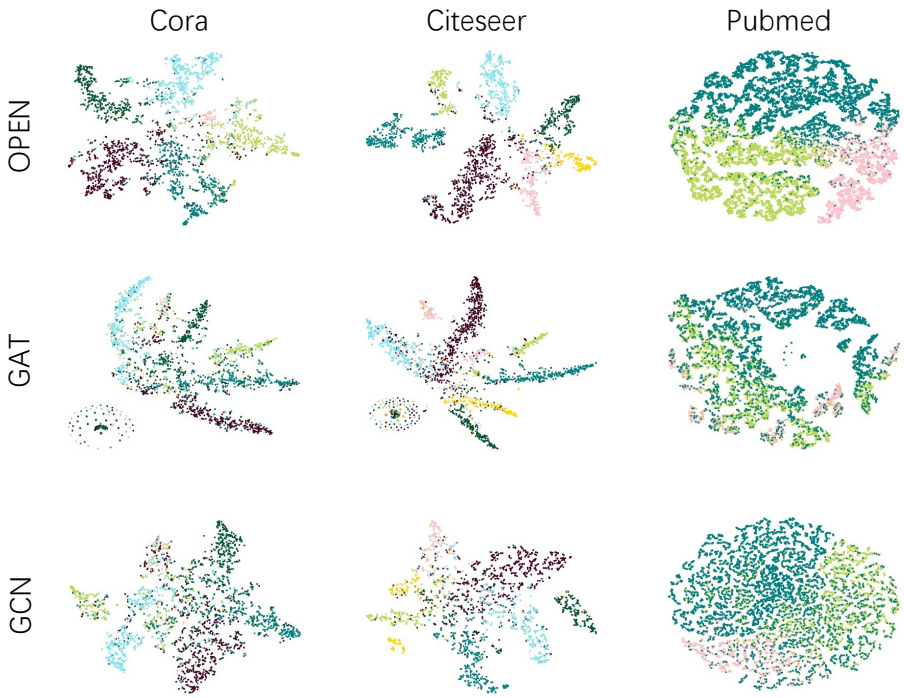

Figure 5: The visualizations of the embeddings obtained from GCN, GAT and OPEN.

B Additional Experimental Results

This section provides additional experimental results to verify some statements and the superiority of the proposed OPEN.

B.1 Node Classification on Another Split

Firstly, to demonstrate the superior performance of the proposed OPEN. Node classification results based on another split, where 20 labelled nodes per class are employed as the training as in GPRGNN [24]. The results are given in Tab. 3. We obtain the similar conclusion as in the main body. Especially, OPEN achieves remarkable improvements on large networks, such as Amazon-Computer and Coauthor-CS. Since the overfitting issue may be more serious on large networks with little nodes labelled, this demonstratesx the capability of OPEN on preventing over-smoothing issue.

B.2 Embedding Visualization

To provide intuitive interpretation, the t-SNE visualizations of node embeddings obtained from GCN, GAT and OPEN are given in Fig. 5. The regions of embeddings of nodes from different classes are with different colors. The shapes of these regions reflect the characteristics of the corresponding methods. The GCN's regions of the embeddings for different classes are overlapped. Thus, the GCN model tends to be under-fitting. The GAT's and OPEN's regions of the embeddings for different classes are distinct. The regions of the embeddings from GAT are very sharp. It indicates that labelled data plays a very essential rule on the embedding, which tends to be overfitting. In contrary, the regions of the embeddings from OPEN are regular. It indicates that the graph topology, which induces smoothing effect, play more important role than the labels., and thus can prevent overfitting. Besides, the results from OPEN are much better than those from GAT on pubmed dataset. GAT's embeddings of nodes with pink color are overlapped from nodes of other colors, while OPEN's embeddings of nodes with pink color are distinct from nodes of other colors. This illustrates the effectiveness of the proposed OPEN compared to GCN and GAT.

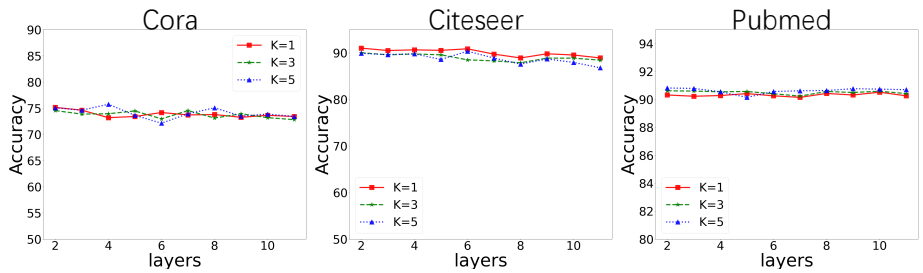

Figure 6: Node classification performances with different numbers of both channels and layers

Table 4: Running Time Comparison (in seconds).

dataset	Cora	Citeseer	Pubmed	Computer	Photo	CS	Physics
GCN	9.89	6.23	5.32	16.8	6.59	19.2	21.58
GAT	10.45	49.31	12.85	95.23	42.11	106.06	201.79
OPEN-Weight	2.61	8.93	3.05	18.46	9.22	18.39	52.94
OPEN-Propagation	10.11	36.09	12.8	65.36	35.62	88.81	149.37
OPEN-layer	271.11	929.09	317.8	1911.36	957.62	1927.81	5443.37
OPEN	12.72	45.02	15.85	83.82	44.84	107.2	202.31

B.3 Ablation Study on Prevent Over-smoothing Issue

Section 4.6 demonstrates the capability of OPEN on preventing over-smoothing issue. DMP [17] proves that the diverse multi-channel propagations provide potentials to prevent over-smoothing issue To investigate which component is more important, Ego-Network modeling or Orthogonal Propagation, ablation study is performed in this section. To this end, the node classification performances with different numbers of channels K and different numbers of layers are given in Fig. 6. It can be observed that OPEN with different K can prevent oversmoothing issue. Thus, ego-network modeling component, which is equivalent to OPEN with $K = 1$, can prevent over-smoothing issue. Note that it is hard to only employ orthogonal propagation without ego-network modeling. Thus, we do not experimentally evaluate the effectiveness of this component. DMP [17] shows that the diversity of the channels promote the ability on preventing over-smoothing issue. Essentially, the orthogonality in multi-channels in OPEN realize the requirement on diversity. Therefore, both two components can prevent over-smoothing issue.

B.4 Running Time Comparisons

To demonstrate the efficiency of the proposed OPEN, the running time comparisons are given in Table. 4. OPEN-Weight and OPEN-Propagation are the weight calculation and propagation parts of the OPEN. Compared to OPEN, where weights are fixed, OPEN-layer denote the variant which learns weights in each layer. The results show that OPEN has the similar running time as GAT. The running time of GAT and OPEN is similar. Note that the running time of GAT and OPEN is longer than that of GCN, due to their multiple-channel propagations and combinations. Besides, while the weight calculation is efficient compared to propagation, the afford of layer-wise weight calculation, i.e., OPEN-layer, is too high. These meet our complexity analysis in Section 3.1.

C Algorithm Description of OPEN

To make the OPEN easy to follow, algorithms description are given. Table. 5 provides the ego-network modeling algorithm, whose output is the propagation weights. Table. 6 is the algorithm of OPEN-layer, which performs the ego-network modeling in each layer. Table. 7 is the final OPEN algorithm, where the ego-network modeling is only performed once.

Table 5: Algorithm-1: Ego-network modeling via PCA

Algorithm-1: Ego-network modeling via PCA

Input: Ego-network of node v: $\mathbf{H}_v = \{\mathbf{h}_{.,1}, \mathbf{h}_{.,2}, \ldots, \mathbf{h}_{.,F}\} \in \mathbf{R}^{|N_v| \times F}$
Output: J eigenvectors of the ego-network of node v: $\{\mathbf{u}_{1,v}, \mathbf{u}_{2,v}, \ldots, \mathbf{u}_{J,v}\}$

Step1: Calculate covariance matrix of $\mathbf{H}_v$:$\mathbf{S}_v = 1/F \sum_{j=1}^{F} (\mathbf{h}_{.,j} - \overline{\mathbf{h}})(\mathbf{h}_{.,j} - \overline{\mathbf{h}})^T$;
Step2: Calculate eigenvectors of covariance matrix $\mathbf{S}_v$ via **Eq**. 6;
Step3: Generate top J eigenvectors:$\{\mathbf{u}_{1,v}, \mathbf{u}_{2,v}, \ldots, \mathbf{u}_{J,v}\}$ via sorting λ_j in descending order;
return $\{\mathbf{u}_{1,v}, \mathbf{u}_{2,v}, \ldots, \mathbf{u}_{J,v}\}$.

Table 6: OPEN-layer

Algorithm-2: OPEN-layer

Input: Feature matrix $\mathbf{X} \in \mathbf{R}^{N \times F}$, Adjacency matrix $\mathbf{A} \in \mathbf{R}^{N \times N}$
Output: Node representations: $\mathbf{H}$

for l=1 to L **do**
  for v=1 to N **do**
    % Ego-network modeling %
    Generate ego-network of v from $\{\mathbf{H}^{(l-1)}, \mathbf{A}\}$: $\mathbf{H}_v^{(l-1)}$;
    % Generate propagation weights for l-th layer %
    Generate propagation weights of v:$\{\mathbf{u}_{1,v}^{(l-1)}, \mathbf{u}_{2,v}^{(l-1)}, \ldots, \mathbf{u}_{J,v}^{(l-1)}\}$ via Algorithm-1 on $\mathbf{H}_v^{(l-1)}$;
    % Orthogonal propagations in multi-channels %
    Update representation of v of channel j: $\mathbf{h}_{j,v}^{(l)\,T} = \mathbf{u}_{j,v}^{(l-1)\,T} \mathbf{H}_v^{(l-1)}, j = 1, 2, \ldots, J$;
    Update representation of v from different channels via **Eq**. 10;
  end for
end for
return $\mathbf{H}^{(L)} = \{\mathbf{h}_1^{(L)}, \mathbf{h}_2^{(L)}, \ldots, \mathbf{h}_N^{(L)}\}$.

Table 7: OPEN

Algorithm-3: OPEN

Input: Feature matrix $\mathbf{X} \in \mathbf{R}^{N \times F}$, Adjacency matrix $\mathbf{A} \in \mathbf{R}^{N \times N}$
Output: Node representations: $\mathbf{H}$

for v=1 to N **do**
  % Ego-network modeling %
  Generate ego-network of v from $\{\mathbf{X}, \mathbf{A}\}$: $\mathbf{H}_v$;
  % Generate propagation weights for all layers %
  Generate propagation weights of v: $\{\mathbf{u}_{1,v}, \mathbf{u}_{2,v}, \ldots, \mathbf{u}_{J,v}\}$ via Algorithm-1 on $\mathbf{H}_v$;
end for
for l=1 to L **do**
  for v=1 to N **do**
    % Ego-network modeling %
    Generate ego-network of v from $\{\mathbf{H}^{(l-1)}, \mathbf{A}\}$: $\mathbf{H}_v^{(l-1)}$;
    % Orthogonal propagations in multi-channels %
    Update representation of v of channel j: $\mathbf{h}_{j,v}^{(l)\,T} = \mathbf{u}_{j,v}^{T} \mathbf{H}_v^{(l-1)}, j = 1, 2, \ldots, J$;
    Update representation of v from different channels via **Eq**. 10;
  end for
end for
return $\mathbf{H}^{(L)} = \{\mathbf{h}_1^{(L)}, \mathbf{h}_2^{(L)}, \ldots, \mathbf{h}_N^{(L)}\}$.