

Graph Decipher: A transparent dual-attention graph neural network to understand the message-passing mechanism for the node classification

Yan Pang¹  | Teng Huang¹ | Zhen Wang²  |
Jianwei Li³  | Poorya Hosseini⁴  | Ji Zhang⁵  |
Chao Liu⁶  | Shan Ai¹

¹Institute of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou, Guangdong, China

²Research Center for Big Data Intelligence, Zhejiang Lab, Hangzhou, Zhejiang, China

³Department of Computer Science, San Jose State University, San Jose, California, USA

⁴Applied Physics Laboratory, Johns Hopkins University, El Segundo, California, USA

⁵School of Mathematics, Physics and Computing, University of Southern Queensland, Toowoomba, Queensland, Australia

⁶Department of Electrical Engineering, University of Colorado Denver, Denver, Colorado, USA

Correspondence

Chao Liu, Department of Electrical Engineering, University of Colorado Denver, Denver, CO 80204, USA.
Email: chao.liu@ucdenver.edu

Abstract

Graph neural networks (GNNs) can be effectively applied to solve many real-world problems across widely diverse fields. Their success is inseparable from the message-passing mechanisms evolving over the years. However, current mechanisms treat all node features equally at the macro-level (node-level), and the optimal aggregation method has not yet been explored. In this paper, we propose a new GNN called Graph Decipher (GD), which transparentizes the message flows of node features from micro-level (feature-level) to global-level and boosts the performance on node classification tasks. Besides, to reduce the computational burden caused by investigating message-passing, only the relevant representative node attributes are extracted by graph feature filters, allowing calculations to be performed in a category-oriented manner. Experiments on 10 node classification data sets show that GD achieves state-of-the-art performance while imposing a substantially lower computational cost. Additionally, since GD has the ability to explore the representative node attributes

by category, it can also be applied to imbalanced node classification on multiclass graph data sets.

KEYWORDS

category-oriented, data augmentation, graph neural network, message-passing mechanism

1 | INTRODUCTION

Graph neural networks (GNNs) offer effective graph-based techniques applied to solve abundant real-world problems in diverse fields, such as social science,^{1,2} remote sensing,³ protein–protein interaction networks,⁴ brain neuroscience,⁵ knowledge graphs,^{6,7} image processing,^{8,9} physical systems,^{10,11} edge computing,^{12,13} information safety,¹⁴ big data,^{15–17} and so forth. The power of current GNNs^{18,19} is largely due to the message-passing mechanism, which recursively aggregates information along edges and updates these newly incorporated features on the center node.

In some early research,^{20,21} messages were passed along edges uniformly without accounting for the priority of either graph structure or node attributes. However, it would be more reasonable for each neighbor node's impact to be distinctive to the central node. Thus, attention-based GNNs^{18,19,22} were proposed to evaluate further how the contribution of neighbors to the central node varies according to the graph characteristics. However, since node attributes are updated by aggregating the received features from neighbors, the impact of each attribute of the nodes on information transmission should not be treated equally as well. Therefore, we hope to transparentize the message flows and make more reasonable use of the graph structure and node attributes. Moreover, particularly in this study, a Graph Decipher (GD) is proposed to account for the impact of these two components on the node classification tasks.

The proposed GD scrutinizes the message-passing mechanism on the graph from three different perspectives: micro-level (feature-level), macro-level (graph-level), and global-level. As shown in Figure 1, a single head of GD contains two parallel branches: the feature attention branch (FAB) explores the node attributes at the micro-level and clarifies their different contributions; while the node attention branch (NAB) focuses on analyzing the difference in graph structure at the macro-level. Then at the global-level, a multihead attention scheme is used to repeat the computations in parallel and then combined to produce a final decision.

For more details, the relevance of node attributes or features is first considered at FAB to gain deeper insights into the message-passing mechanism. Graph feature pooling (GFP) and upsampling modules are introduced to update the node feature matrix according to node category. To estimate the impact of each node's internal characteristics on the node classification task, a dimension-based self-attention mechanism is proposed, which exploits the attention granted to the node attributes in graph learning. This innovative procedure yields significant improvements in finding the respective optimal attributes of each node according to the categories. Moreover, focusing only on the optimal attributes instead of all helps reduce the computational burden. Then at the end of the FAB, its output is combined with the concurrent NAB branch that strengthens the contribution of neighbor nodes to the central node for further calculations. Ultimately, a multihead attention scheme that consists of multiple parallel single-heads outputs the final decisions. Experiments show this proposed mechanism significantly outperforms the other state-of-the-art (SOTA) work on 10 common graph data sets used for

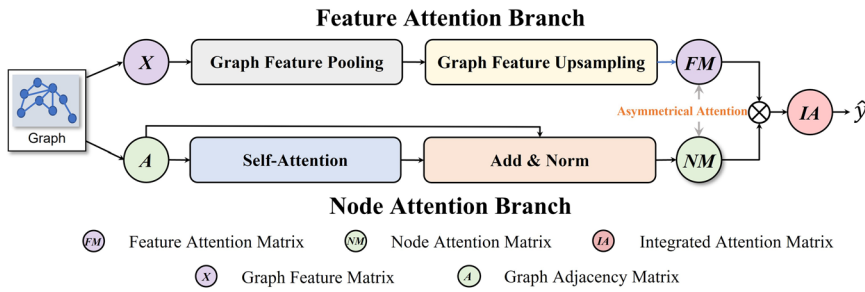


FIGURE 1 A single-head Graph Decipherer (GD) includes two main attention branches: the feature attention branch (FAB) and the node attention branch (NAB). While the NAB (bottom) determines the priority of nodes under the node classification task at the graph-level, the FAB (top) is dedicated to deep exploiting the importance of node attributes in a category-oriented manner at the feature-level. The FAB is composed of graph feature pooling (GFP) and graph feature upsampling (GFU) modules (zoomed-in Figure 2). The objective of the GFU module is to obtain the internal priority of the prominent representative features highlighted by category in the GFP. [Color figure can be viewed at wileyonlinelibrary.com]

node classification tasks: Cora,²³ Citeseer,²³ PubMed,²⁴ Amazon Computers and Photo,²⁵ Coauthor CS and Physics,²⁶ Cornell and Texas,²⁷ and Chameleon.²⁸

GD also performs well on the imbalanced graph data sets, whose distribution of node categories is inhomogeneous. Samples from a minority of categories are usually under-represented, resulting in suboptimal performance. Our solution is to increase the number of nodes in the minorities to balance the distribution. Only the dominant node attributes of each category analyzed by GD are retained, and the unrepresentative features are randomly dropped or replaced during reproduction.

The contributions in this paper are summarized as follows:

- A transparent GNN, GD, is proposed. This scheme can explain how the graph structure and node attributes affect message-passing on the node classification tasks from the three levels of macro, micro, and global.
- Unlike the common methods that assign the same weight to each feature of the same node, we design novel GFP and upsampling modules to extract and pay more attention to the dominant features for optimizing the message-passing mechanism.
- To reduce the computational burden, we analyze node attributes in groups by category, and only the representative attributes extracted by the GFP filter are utilized in the calculation.
- Since GD has the ability to perform representative analysis on the features of each node, it can be used to augment the samples from a minority of categories, thereby improving the performance on the imbalanced node classification tasks.

2 | RELATED WORK

2.1 | Graph neural networks

Recent researches^{29,30} applied aggregation operations directly to graphs and aggregated messages with shared weights from neighbors to each center node. These GNNs only considered passing messages uniformly from one- or two-hop neighbors along edges.

GraphSAGE¹ aggregated and updated features in a range of the two-hop neighbors to the center node. Directed graph convolutional network²⁰ considered the first- and second-order proximity to aggregate the attributes on the directed graphs. However, these message aggregators collected information from all neighbors equally, ignoring the relative importance of different neighbor nodes.

To solve the issue mentioned above, more studies^{19,31,32} considered the attention-based architecture to compute the hidden representations of each node on the graph. By calculating the node attention coefficients from the neighbors before reaching the center node, graph attention network (GAT)¹⁹ implicitly specified the relevance of neighbor nodes in the node classification task. Gated attention networks³¹ considered the priority of the multihead attention with a convolution subnetwork. Graph convolutional networks (GCN)³³ utilized weighted structural features to explore directional structural information for nodes. However, these GNNs only concentrated at the macro-level by assigning arbitrary weights to the neighbor nodes. Although the neighbor nodes' priority was investigated, the relevance of node features was still ignored. For example, since all attributes of a node share the same weight, at nodes marked as lower weights, significant internal attributes are suppressed to propagate. Conversely, the nodes with higher attention coefficients may have inconsequential attributes amplified and passed along to the central node, which causes information interference that limits the overall accuracy of the network.

Our proposed GD exploits deep characteristics of the message-passing mechanism on both graph structure and node attributes. The transparent mechanism allows straightforward investigation of each node's internal and external impacts on the node classification tasks. And it can also be utilized in a wide range of applications, such as social networks, recommended systems, the internet of things, emotion estimation, and so forth.

2.2 | Graph data augmentation

Node classification is a primary graph task for a wide range of applications.^{5,6,10} Many researches^{34,35} have demonstrated that neural networks are more inclined to learn features from categories with larger amounts of data, which results in relatively lower accuracy of minor categories.

Rong et al.³⁶ designed a DropEdge technology to randomly delete edges before each training epoch to prevent messages from being passed from the nodes labeled as the majority category. Chen³⁷ proposed to change the connections between nodes by adding edges to nodes of the same category or disconnecting nodes from different categories. Although the data-imbalance issue can be alleviated, this approach may lead to propagation errors on the modified graph. Shi³⁵ facilitated the partition of the annotated nodes with a class-conditioned adversarial strategy. However, the number of nodes of the minorities is not increased, and the inconsequential attributes may increase the difficulty of model training.

We employ GD to augment the samples from a minority of categories by performing representative analysis on node attributes to solve the above issues. The dominant and representative node attributes are amplified in the minorities, while the inconsequential ones are suppressed after the data augmentation. Experiments show that such a method significantly improves the performance of the minorities on the imbalanced node classification tasks.

3 | PRELIMINARIES

Definition 1 (General graph concept). In general, a graph, G , contains two main matrices: adjacency matrix, A , for graph structure, and feature matrix, \mathbf{X} , for graph attributes. The element of A indicates the connections (Edges) among objects (Nodes). The row of \mathbf{X} represents the feature representation of each node in the graph.

Definition 2 (Graph node classification). Given a graph (A and \mathbf{X}), the eventual task is to estimate the anonymous label y of the node $v \subseteq N$ by aggregating and updating the messages from its neighbors.

4 | METHODS

To investigate the potential of graphs on the node classification task, NAB and FAB are explored to track down the message-passing mechanism in parallel, as shown in Figure 1. The NAB learns a node attention matrix, \mathbf{NM} , which represents the contribution of neighbors to the central node at the macro-level. While an innovative FAB is utilized to obtain the feature attention matrix (\mathbf{FM}) to emphasize each node's attributes at the micro-level. Then, these two matrices are combined to form an integrated attention matrix (\mathbf{IA}), which contains both attention information of the graph structure and node attributes to complete a single-head prediction. The multihead mechanism is finally used to stabilize the learning process of the node classification task.

4.1 | Macro-level: Node attention branch

An adjacency matrix is usually used in graph-related tasks to represent the relationship among nodes on a graph. But it ignores the fact that neighbors may contribute differently to the central node. And we hope that messages from important neighbors could get more attention when they converge to the central node. Therefore, in this study, inspired by GATs, an NAB is used to calculate the contribution of neighbors to the central node according to the characteristics of the graph task, thereby assigning different weights to the message-passing flows.

Similar to GAT, a node self-attention matrix, β_{vs} , is learned to determine the relevance between neighbors and the center node, v , as shown in Equation (1).

$$\beta_{vs} = \text{softmax}(e_{vs}), \quad (1)$$

where $e_{vs} = \gamma(W \cdot \mathbf{x}_v, W \cdot \mathbf{x}_s)$ represents the importance of neighbor nodes s to v . γ indicates the self-attention mechanism, and W is the weight matrix. x_v and x_s are node attributes of nodes v and s . Once attention coefficients are calculated, the activation function σ is applied to get the final nonlinear node attributes in Equation (2).

$$NAB : \mathbf{x}_v = \sigma \left(\sum_{s \subseteq N_v} \beta_{vs} W \cdot \mathbf{x}_s \right). \quad (2)$$

The node attention coefficients represent the contributions of neighbors to the center node in NAB. Yet, the roles played by the different internal attributes of each node have not been considered. Therefore, a new FAB is added to interact with NAB to update the attention of nodes and their corresponding attributes.

4.2 | Micro-level: Feature attention branch

In the message-passing mechanism of classical GNNs, node attributes are usually refined through two steps: gathering and updating. In both, information is processed uniformly on all feature dimensions, which does not factor in the significance of node attributes. It may lead that inconsequential information causing redundancy or restricting the desired attributes. Thus, it is crucial to evaluate the internal priority of all node attributes, allowing representative features to be examined more closely in the message-passing procedures. However, this would cause a steep computation burden.

We propose a category-oriented self-attention manner to determine the internal priority of attributes and reduce the number of computations. This manner also improves the efficiency of information transmission and the network's performance under the node classification task. Two modules connected in series constitute the FAB: GFP and graph feature upsampling (GFU). The first module highlights prominent representative attributes and reduces the size of feature maps; the second module distinguishes the priority of these attributes by categories.

4.2.1 | GFP module

Since we hope to encourage the representative attributes and restrict the inconsequential ones in the message-passing procedures, an intuitive method is to calculate the relevance of all node attributes. In this case, the computation burden is equal to $O(\mathbf{X}^N)$, where N is the amount of nodes. This idealized approach can only be applied on the small graph because the computation burden may be much heavier as the number of nodes on a graph increases. However, in practice, we often encounter larger graphs with substantial nodes and edges in areas, such as social network graphs.

It is a big challenge to balance the demands of investigation and categorize computations under node classification tasks. From the graph perspective, nodes are categorized together if they share similar attributes. Inspired by this, we explore node attributes using category-oriented feature attention coefficients under the graph task. Inner each category, the most significant attributes can be sorted, selected, and then sent along graph edges, while the less significant attributes are suppressed. In this manner, the computation burden is mainly derived from the number of node categories, since the focus is given to prioritizing attributes by category instead of individual nodes. Each feature dimension shares the same attention coefficient by node category, whose number is finite and far lower than the total number of nodes, even on huge graphs. Thus, the computation can be reduced sharply to $O(\mathbf{X})$ in the learning process, because the total sizes of all concatenated attention matrices labeled by node category are equal to the size of the feature matrix, \mathbf{X} .

In a category-oriented manner, nodes are assigned into finite groups according to their categories. As shown in Figure 2, the original 2D feature matrix \mathbf{X} is divided into C subfeature matrices. The goal of GFP module is to highlight the local prominent attributes by category. It contains three main steps: sort the nodes in the two-dimensional (2D) subfeature matrix, transform the sorted matrices to 3D subfeature maps, and highlight the local dominant attributes by category.

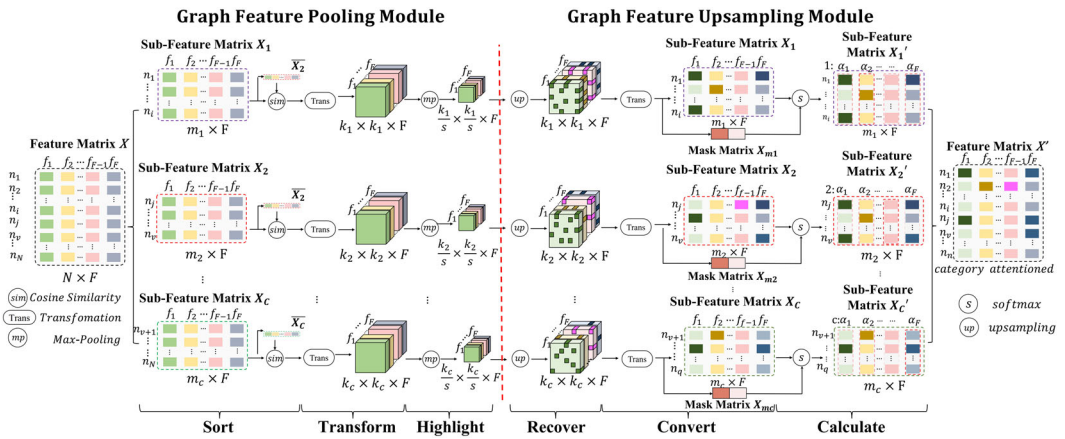


FIGURE 2 Overview of the FAB. It includes two connected serial modules, GFP and GFU, separated by a red dash line. Each module contains three main operations. The details are illustrated in Algorithm 1. FAB, feature attention branch; GFP, graph feature pooling; GFU, graph feature upsampling. [Color figure can be viewed at wileyonlinelibrary.com]

Sort: Since nodes in the same group or submatrix have comparable attributes, the mean feature vector can be regarded as the most representative attribute vector to indicate the corresponding category in each subfeature matrix. Then, nodes are sorted in each subfeature matrix in Equation (3) based on the similarity between each node attributes with the mean feature vector, $\bar{\mathbf{x}}_i = \frac{1}{m_i} \sum_{l=1}^{m_i} \mathbf{x}_l$.

$$sim(\bar{\mathbf{x}}_i, \mathbf{x}) = \cos \frac{\bar{\mathbf{x}}_i \odot \mathbf{x}}{\|\bar{\mathbf{x}}_i\| \cdot \|\mathbf{x}\|}, \tag{3}$$

where $\bar{\mathbf{x}}_i$ is the mean feature vector, and m_i is the number of feature vectors of the i th subfeature matrix. \mathbf{x} indicates each feature vector of the subfeature matrix, and \odot represents the dot product between two feature vectors.

Transform: Since adjacent nodes represent similar attributes in the sorted subfeature matrices, we hope to identify the local dominant values, as they indicate the most representative attributes by dimensions, F . In this procedure, each 2D matrix $m_i \times F$ is transformed to a corresponding 3D feature map $k_i \times k_i \times F$ labeled with the node category, where F is the depth of the 3D feature map and also the dimension of the 2D feature matrix. Note that the square root is not necessarily an integer, thus k_i is rounded up to the nearest integer, $k_i = \lceil \sqrt{m_i} \rceil$. In other words, the shape of the original 2D subfeature matrix m_i is increased to a larger value k_i^2 in most cases.

Highlight: The final step of GFP is to highlight the local dominant and representative attributes in the max-pooling operation on each feature map with the corresponding category. The stride of the operation is equal to the size of the pooling filter, and the dimension of the updated feature maps is given by Equation (4).

$$dim(feat_maps) = \left(\frac{k_i}{s}, \frac{k_i}{s}, F \right). \tag{4}$$

The stride, s , determines the number of representative attributes utilized to calculate the interior priority of nodes in GFU. The computation burden and performance must be balanced, as large

stride values lower the computation burden while degrading performance, while lower stride values induce prohibitively enormous computation burdens. This is discussed further in Appendix A.

4.2.2 | GFU module

Although the local dominant representative node attributes in each category are highlighted at the end of the GFP, the internal priority of the representative features is not yet known. Thus, the objective of the serially connected module, GFU, is to obtain the internal priority of the locally prominent and representative features in distinctive categories, as illustrated in Figure 2. GFU includes three main steps: recover the 3D subfeature maps, convert them back to the 2D subfeature matrix, and calculate the inner priority of dominant attributes.

Recover: In the GFU, the upsampling operation is performed first to recover feature maps from the size $(\frac{k_i}{s}, \frac{k_i}{s}, F)$ to (k_i, k_i, F) . Since the total size of the recovered feature map is extensive, vacant positions in the matrix are filled with zeros as they do not affect the graph semantics in corresponding feature dimensions.

Convert: Next, a transformation operation is performed to convert each 3D feature map back to a 2D subfeature matrix \mathbf{X}' . Because nodes are assigned by the feature similarity in the GFP module, they are sorted in their original order in each subfeature matrix. Meanwhile, a corresponding mask with the same matrix shape with a special value (0/1) is also generated to record the position of the local representative features. A value of 1 in each mask indicates a local representative attribute at this position in the corresponding subfeature matrix. In contrast, a value of 0 representing this position's corresponding value may be unrepresentative in this category.

Calculate: Since the local representative attributes have already been determined in each updated feature matrix, they can now be utilized to find the internal priority of nodes by category. This step applies a learnable self-attention scheme to each subfeature matrix in Equation (5).

$$\alpha_j = \text{softmax}(\mathbf{x}_j) = \frac{\exp(\mathbf{x}_j)}{\sum_{j \in \mathcal{C}_f} \exp(\mathbf{x}_j)}, \quad (5)$$

Algorithm 1: The dimension-based self-attention mechanism on the feature attention branch

Input : The original 2D feature matrix \mathbf{X}

Output: The updated 2D feature matrix \mathbf{X}'

1. **Split** \mathbf{X} to n sub-feature matrices // $n = C + 1$

// for each category

2. **for** $i \leftarrow 1$ to $n - 1$ **do**

// GFP module

3. Mean feature vector: $\bar{\mathbf{x}}_i = \frac{1}{m_i} \sum_1^{m_i} \mathbf{x}$

4. Similarity: $\text{sim}(\bar{\mathbf{x}}_i, \mathbf{x}) = \cos \frac{\bar{\mathbf{x}}_i \odot \mathbf{x}}{\|\bar{\mathbf{x}}_i\| \cdot \|\mathbf{x}\|}$

5. **Sort** the nodes based on the sim

6. **Transform** to 3D feature maps

$(m_i \times F) \mapsto (k_i \times k_i \times F)$ where $k_i = \lceil \sqrt{m_i} \rceil$

7. **Highlight** $(k_i \times k_i \times F) \mapsto (\frac{k_i}{s} \times \frac{k_i}{s} \times F)$

// GFU module

8. **Recover** $(\frac{k_i}{s} \times \frac{k_i}{s} \times F) \mapsto (k_i \times k_i \times F)$

9. **Convert** back to 2D sub-feature matrix

$(k_i \times k_i \times F) \mapsto (m_i \times F)$

10. Update the mask for the representative features

11. **Calculate** Feature Attention Coefficient with mask

$\alpha_j = \text{softmax}(\mathbf{x}_j)$ where $j \leftarrow (1, F)$

end

12. **Merge** all sub-feature matrices to the matrix \mathbf{X}'

where \mathbf{x}_j indicates the vector of the j th dimension in each subfeature matrix. Since only the local dominant and representative features are needed in this step, the computational burden can be further reduced by using masks that record the position of local representative features, allowing the unnecessary zeros to be ignored in the computation. The new nonlinear node v 's feature is applied by the activation function ϕ in Equation (6).

$$FAB : \mathbf{x}_v = \phi \left(\sum_{j \subseteq (1,F)} \alpha_j W \mathbf{x}_v \right). \quad (6)$$

The final step is to capture both attention matrices from parallel branches in Equation (7).

$$\mathbf{h} = \phi \left[W \cdot \sigma \left(\sum_{s \subseteq N_v} \beta_{vs} W \mathbf{x}_s \right) \cdot \phi \left(\sum_{j \subseteq (1,F)} \alpha_j W \mathbf{x}_v \right) \right]. \quad (7)$$

This combined attention matrix contains the graph structure's attention information and the nodes' internal attributes needed to complete a single-head prediction. The algorithm of FAB is summarized in Algorithm 1.

4.3 | Global-level: Multihead attention mechanism

The multihead attention mechanism assigns distinctive attentions, \mathbf{c} , on each head to stabilize the learning process. Thus the embedding at the next hidden layer, $\mathbf{h}_{l+1} = \sigma \left(\frac{1}{\zeta} \sum_{\zeta=1}^{\zeta} \mathbf{c} \cdot \mathbf{h}_l \right)$, is updated, where ζ is the head number.

5 | EXPERIMENTS

5.1 | Performance on node classification task

5.1.1 | Comparison with basic frameworks

We first experimentally validate our proposed algorithm on three graph citation data sets (Cora, Citeseer, and PubMed), two copurchase data sets (Amazon Computers and Amazon Photo), and two Coauthor graphs (Coauthor CS and Coauthor Physics). More details about data sets are in Appendix A.

The details of the experiment's setup are introduced in Appendix A. The accuracy for the node classification task of distinctive algorithms on all seven data sets is illustrated in Table 1. It can be observed that the performance of the GNNs (MoNet,³⁸ GCN, GraphSage, GAT, and ours) surpasses the performance of non-GNN frameworks (MLP and LabelProp³⁹), which benefit from the message-passing mechanism by considering both node attributes and graph structure on a graph.

The MoNet, GCN, and GraphSage pass messages along edges uniformly on the graph. GAT evaluates only the contribution of direct neighbors to the central node on the graph structure.

TABLE 1 Comparison of the performance (accuracy%) of different algorithms for the node classification task on different data sets

Method	Cora	Citeseer	PubMed	Coauthor CS	Coauthor Physics	Amazon Computers	Amazon Photo	Year
MLP ¹⁹	55.1	46.5	71.4	88.3	88.9	45.1	69.6	2017
LabelProp ³⁹	73.9	66.7	72.3	76.7	86.8	75.0	83.9	2009
MoNet ³⁸	81.7	71.2	78.6	90.8	92.5	83.5	91.2	2017
GCN ⁴⁰	81.5	70.3	79.0	91.1	92.8	82.6	91.2	2016
GraphSAGE ¹	79.2	71.6	77.4	91.3	93.0	82.4	91.4	2017
GAT ¹⁹	83.4	72.5	79.0	90.5	92.5	78.0	85.1	2017
GD (ours)	88.3	78.9	85.5	95.9	97.3	83.1	89.3	2022

Note: Bold values indicate the optimal choice by trading off the complexity and average precision.

Abbreviations: GAT, graph attention network; GCN, graph convolutional network; GD, Graph Decipher; MLP, multilayer perceptron.

The measured performance of GAT is 83.4%, 72.5%, and 79.0% on the Cora, Citeseer, and PubMed, respectively. It is superior to uniform GNNs on small graph data sets, but it degrades on Amazon Computers and Amazon Photos, with extended node attributes. Since all attributes are considered uniformly, the desired attributes are inefficiently confined to enhance performance.

Because GD effectively biases its attention to representative node attributes alongside the most relevant neighbor nodes on the graph structure, it achieves SOTA performance in five out of seven data sets under the node classification task. GD achieves the accuracy of 88.3%, 78.9%, 85.5%, 95.9%, and 97.3% on the Cora, Citeseer, PubMed, Coauthor CS, and Coauthor Physics data sets, respectively. In the case of the two most extensive data sets, Amazon Computer and Amazon Photo, GD lagged behind the front-runner by only 0.4% and 2.1%, respectively. These experiments demonstrate the contribution of high-priority components, consisting of node attributes and neighbors of the graph structure, under the node classification task.

The size of the pooling filter in GFP is a significant parameter that affects GD's performance and computation burden as it determines the amount of local dominant and representative features by category in the learning process. In experiments, the filter size, which equals 2, typically imposes a slightly more significant computation burden than competing algorithms. More details about it and other ablation studies are discussed in Appendix A.

5.1.2 | Comparison with more SOTA approaches

Inspired by the results of previous experiments, we hope to demonstrate the power of GD by comparing it with more SOTA approaches to those data sets. Table 2 illustrates the performance of all SOTA approaches under the node classification task on three graph citation data sets and two Coauthor graphs.

GraphStar merges message-passing relay and self-attention mechanism to classify the node embedding on the graph.⁴¹ As a reinforcement learning-based approach, GraphNAS automatically utilizes the best recurrent network to maximize the expected accuracy in the test procedure.⁴² Graph-Bert only employs a self-attention mechanism instead of graph

TABLE 2 Comparison of more state-of-the-art node classification approaches on Cora, Citeseer, PubMed, Coauthor CS, and Coauthor Physics data sets

Method	Cora	Citeseer	PubMed	Coauthor CS	Coauthor Physics	Year
GraphStar ⁴¹	82.1	71.0	77.2	–	–	2019
GraphNAS ⁴²	84.2	73.1	79.6	–	–	2019
Graph-Bert ⁴³	84.3	71.2	79.3	–	–	2020
DifNet ⁴⁴	85.1	72.7	79.5	–	–	2020
Cleora ⁴⁵	86.8	75.7	80.2	–	–	2021
LinkDistMLP ⁴⁶	87.6	75.3	88.8	95.7	96.9	2021
CoLinkDist ⁴⁶	87.9	75.8	89.6	95.8	97.1	2021
LinkDist ⁴⁶	88.2	74.7	88.8	95.7	96.9	2021
3ference ⁴⁷	87.8	76.3	88.9	95.9	97.2	2022
GD (ours)	88.3	78.9	85.5	95.9	97.3	2022

Note: The best results (Acc) are shown in bold.

Abbreviation: GD, Graph Decipher.

aggregation operators.⁴³ DIFNET also considers the attention mechanism to diffuse the information of neighborhood nodes along edges.⁴⁴ The performance of attention-based approaches is powerful on graph citation data sets, demonstrating that the attention-based mechanism is efficient in extracting node features at the macro-level on the graph.

Besides the above four attention-based approaches, another five nonattention-based SOTA approaches are also compared. Cleora iterative weighted averaging the neighbor features and then normalizing embedding across dimensions.⁴⁵ LinkDist series (LinkDistMLP, CoLinkDist, and LinkDist) extract useful features by distilling self-knowledge from associated couple nodes.⁴⁶ 3ference analyzes the transition patterns of node labels on the graph.⁴⁷ The performances of these SOTA approaches are improved. However, our proposed GD achieved four out of five top on all graph citation and coauthor data sets. These experiments demonstrate the contribution of high-priority category-oriented node features at the micro-level under the node classification task.

5.1.3 | Comparison with SOTA approaches on more data sets

Besides three graph citation data sets, two copurchase data sets, and two Coauthor graphs, we compare more SOTA approaches on three heterophily data sets, Cornell, Chameleon, and Texas. Cornell and Texas data sets represent hyperlinks (edges) between web pages (nodes) from the department of computer science in different universities. Chameleon data set performs the web pages to discuss corresponding cases in Wikipedia.

In these experiments, we compare the proposed network with another nine SOTA approaches, which enhance the aggregation mechanism for node representation learning. Geom-GCN extracts the structural information from a continuous space underlying the graph aggregation on a graph.⁴⁸ SDRF classifies the nodes by introducing an edge-based combinatorial curvature.⁴⁹ CNMPGNN exploits node patterns by utilizing the common-

TABLE 3 Comparison of more state-of-the-art node classification approaches on Cornell, Chameleon, and Texas data sets

Method	Cornell	Chameleon	Texas	Year
Geom-GCN-S ⁴⁸	55.68	59.96	59.73	2020
Geom-GCN-I ⁴⁸	56.76	60.31	57.58	2020
Geom-GCN-P ⁴⁸	60.81	60.90	67.57	2020
SDRF ⁴⁹	54.61	42.73	64.40	2021
CNMPGNN ⁵⁰	82.38	73.29	85.68	2021
GGCN ⁵¹	85.68	71.14	84.86	2021
FDGATII ⁵²	82.43	65.17	80.54	2022
GloGNN ⁵³	83.51	69.78	84.32	2022
GloGNN++ ⁵³	85.95	71.21	84.05	2022
GD (ours)	87.26	72.47	86.73	2022

Note: The best results (Acc) are shown in bold.

Abbreviations: CNMPGNN, Common Neighbors Motifs Perceptive Graph Neural Network; FDGATII, fast dynamic graph attention with initial residual and identity mapping; GCN, graph convolutional network; GD, Graph Decipher; GGCN, graph convolutional neural network; GNN, graph neural network.

neighbors-based motifs.⁵⁰ GGCN proposes the degree corrections and signed messages to learn the node representations.⁵¹ FDGATII adopts an expressive dynamic self-attention mechanism to improve the performance under node classification task.⁵²

GloGNN and GloGNN++ aggregates representations from global nodes to the center node.⁵³

Illustrated in Table 3, GD receives the accuracy of 87.26%, 72.47%, and 86.73% on Cornell, Chameleon, and Texas data set, respectively. It achieves the SOTA performance in two out of three data sets under the node classification task. Even in the Chameleon data set, GD lagged behind the front-runner by only 0.82%.

5.2 | Graph data augmentation

In Section 2.2, we discuss the significance of graph data augmentation and its influence on the multiclass imbalanced data set. Without data augmentation, most false predictions are concentrated in minority categories in each graph data set, diminishing the network performance. Current research^{34,35,54,55} rely on attempts to balance the distribution of categories by sampling from a portion of the original data sets or class-conditioned adversarial graph learning. The primary issue with these approaches is that all node features or attributes are considered in the graph learning process regardless of their relevance. It means insignificant node attributes are amplified and affect graph learning. We hope to evaluate only the critical node features or attributes to balance the node category distribution. Thus, we propose a new method of graph data augmentation to improve the network's performance, especially in minority categories, by utilizing the FAB of the message-passing mechanism.

5.2.1 | Data augmentation on two categories: Majority and minority

The distribution of node categories of all graph data sets is analyzed in Appendix A. The multiclass imbalanced phenomenon generally causes most false predictions in the node category with the smaller data collection in the inference process. To prove our conjecture, nodes from the major and minor node categories are extracted to create a new imbalanced train and test data sets separately. The network achieves 99.7% and 98.9% accuracy on new imbalanced Cora and Citeseer data sets. We receive this kind of ‘extraordinary’ performance because the node attributes with the major node category are weighted more heavily in the training process and propagated extensively on the imbalanced test data set. However, if the retrained network only tests on the minority, the accuracy drops to 47.6% and 42.1%, respectively, agreeing with our prediction. To solve this issue, we reconsidered the utilization of feature attention of the pre-trained model to gain efficiency. Therefore, we designed a series of experiments to find feasible data augmentation approaches to alleviate the imbalanced problem.

In these experiments, each data set is split into 10 imbalanced data sets, containing only two node categories, majority and minority, with differing proportions. The proportion in the imbalanced data set is around 10:1 at the beginning. A straightforward approach, Originally (ORI), for node reproduction is directly cloning the nodes and node attributes in the minority. However, it may lead to the induced overfitting issue because the network amplifies and learns the inconsequential node features in the learning process. Thus, another innovative approach for node reproduction is proposed.

Since GD can explore the priority of node attributes under the node classification task, the node attributes are separated into two groups: representative and unrepresentative features. The former are retained in the reproduction, and the unrepresentative features are addressed with two different approaches to reproducing the minority: (1) AA: All the inconsequential attributes are cleaned, and the cleaned attributes are directly cloned. (2) AP: Some inconsequential attributes are cleared randomly for each reproduction. Thus, the values of reproduced data are not the same. Finally, the propagation of majority and minority is followed by 10:1, 10:2, 10:3, 10:4, 10:5, 10:6, 10:7, 10:8, 10:9, and 10:10. The GD is then retrained on the newly synthesized data sets.

The test data set now only includes the minority in the inference process. The accuracy of the imbalanced synthesized data sets is summarized in Figure 3. The AA results arrangement is less than 60% in all cases because all the inconsequential attributes are cleared. This result indicates that even inconsequential attributes have insignificant contributions to the node classification tasks. The maximum scores of the ORI approach increased in all cases. However, the scores of the ORI approach are only better than AA in one of our seven cases. While there are performance gains, improvements are still needed if the minority node information is repetitive. Furthermore, the AP approach’s performance is compared with the ORI approach in every case because the distribution of the synthesized data set with AP is diversified.

Following this discovery, another approach called AN is used to reproduce the data set. This approach introduces small amounts of random noise to replace part inconsequential node attributes. An achieved higher scores than AP on the following data sets: Cora, Citeseer, Coauthor CS. However, in the remaining data sets, AP performed better than AN. The interquartile range (IQR) provides a visual indicator regarding the spread of accuracy amongst different synthesized data sets for each case. From Figure 3, the differences in IQRs for both AN and AP are tiny for each case, meaning both approaches can improve predictions on the minority category of the inhomogeneous data set. However, it is essential to note that the

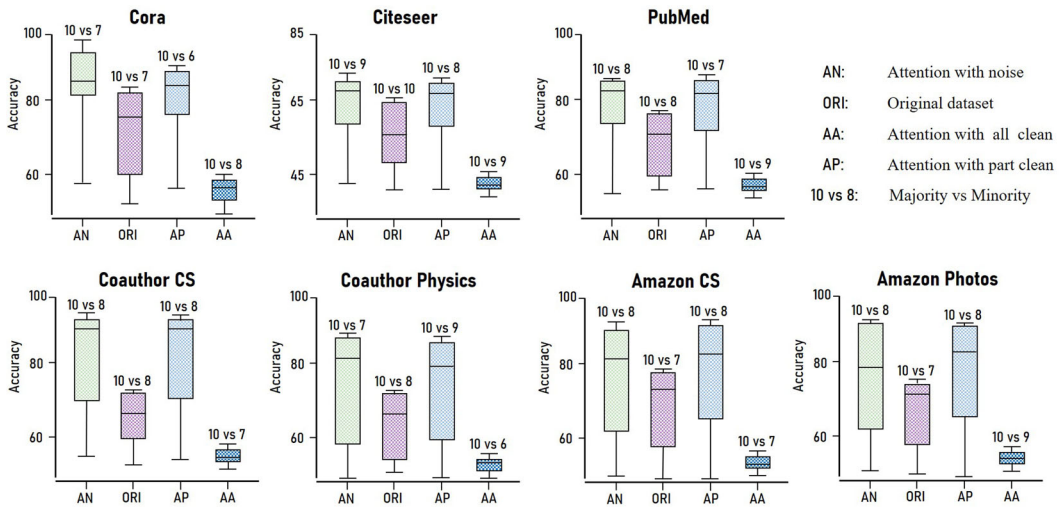


FIGURE 3 Accuracy indicates the effect on the performance of different approaches in the minority category. The numbers above the max bar, 10 versus 7, represent the ideal proportion of the majority and minority categories on each inhomogeneous synthesized data set. AA, Attention with All Clean; AN, Attention with Noise; AP, Attention with Part Clean; Ori, Originally clone the information of minority. [Color figure can be viewed at wileyonlinelibrary.com]

artificial noise introduced has no practical use in real-world applications. Thus, although the network is more robust, the AN approach cannot be used in certain domains which require a strong interpretability graph network, such as in medical science.

In addition, the number above the max bar in Figure 3 represents the proportion of the majority and minority categories on each imbalanced synthesized data set. Here the max scores of each approach are not from the most balanced setting, 10 versus 10, of the synthesized data sets in each case. For example, the best setting of AP in the Cora synthesized case is 10 (majority category) versus 6 (minority category). On the other hand, in synthesized Coauthor CS, the best proportion of the majority and minority categories is 10 versus 8 in both the AN and AP approaches. This is because the synthesized data sets are generated based on a portion of the minority nodes. Thus, an appropriate number of generated nodes from the minority category is sufficient to balance the inhomogeneous multiclass issue.

5.2.2 | Data augmentation on all categories

This section demonstrates GD's performance after data augmentation of all categories under the node classification task. We followed the same procedures in Section 5.2.1 to balance the node distribution of all seven data sets: (1) Analyze all data sets by a pretrained GD; (2) Retain the representative features; (3) Reproduce the minorities by AP. After data augmentation, the ratio of each two categories is concentrated at 1:1.5 on each training data set. Then, our network is retrained on new imbalanced graph data sets. Figure 4 illustrates the increment of the network performance with the data augmentation. As a result, networks' performance is further improved than before augmentation on all 10 data sets, demonstrating the improvements delivered by our innovative feature attention mechanism of GD. Currently,

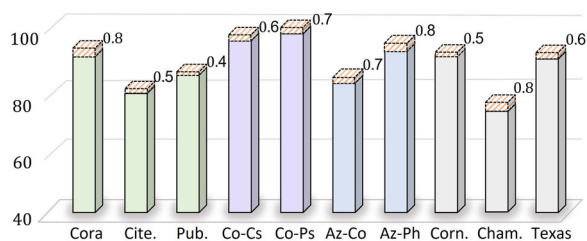


FIGURE 4 Increment histogram. The y-axis indicates the accuracy of GD under the node classification task. The solid bar is the performance of GD on each original data set. The shaded area shows the increment of network performance with the data augmentation approach, AP, on each graph data set. AP, Attention with Part Clean; Az-Cs, Amazon CS; Az-Ph, Amazon Photos; Cham., Chameleon; Cite, Citeseer; Co-Cs, Coauthor CS; Co-Ps, Coauthor Physics; Corn., Cornell; GD, Graph Decipher; Pub., PubMed. [Color figure can be viewed at wileyonlinelibrary.com]

GD fails to handle the time-varied graph-based structure because the nodes and links may emerge and disappear along the time dimension. Our future work will advance the GD to be suitable for dynamic graphs.

6 | CONCLUSION

This paper proposes a transparent GNN, GD, that investigates the message-passing mechanism on a graph under the node classification task. GD improves functionality by showing how the graph structure and node attributes affect the message-passing mechanism in the node classification task from the micro-, macro-, and global-levels. By giving higher priority to both neighbor nodes on the graph structure and representative features of node attributes, GD efficiently improves performance on the 10 graph data sets studied. Meanwhile, the computation burden imposed by GD is acceptable due to three novel features: (i) it explores the node attributes with category-oriented feature attention coefficients; (ii) it investigates the representative attributes retained by the GFP filter; (iii) it calculates the interior priority of node attributes on the sparse matrix generated from the mask. Additionally, an innovative GD-based graph data augmentation approach alleviates the imbalanced node classification problem on multiclass graph data sets. We hope these discoveries will encourage future research into the possibilities of GNNs in additional real-world applications. In the future, we hope to advance and transfer our algorithm to the time-varied graph-based structure, which is more complicated. The current approach fails to handle those dynamic graphs which record all continuous evolution over time. Because links and nodes may emerge and disappear on the graphs along the temporal dimension, it is time-consuming to investigate node representations by category at each moment. Especially, the nodes in one category are disappeared at an occasion.





DATA AVAILABILITY STATEMENT

No. Research data are not shared.

ORCID

Yan Pang  <http://orcid.org/0000-0002-6483-8326>

Zhen Wang  <http://orcid.org/0000-0002-8637-8375>

Jianwei Li  <http://orcid.org/0000-0002-5372-223X>
 Poorya Hosseini  <http://orcid.org/0000-0001-6929-8106>
 Ji Zhang  <http://orcid.org/0000-0001-7167-6970>
 Chao Liu  <http://orcid.org/0000-0002-8703-4232>

REFERENCES

- Hamilton WL, Ying R, Leskovec J. Inductive representation learning on large graphs. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*; 2017:1025-1035.
- Li J, Hu X, Xiong P, et al. The dynamic privacy-preserving mechanisms for online dynamic social networks. *IEEE Trans Knowl Data Eng.* 2020;34(6):2962-2974.
- Ai S, Koe ASV, Huang T. *Adversarial Perturbation in Remote Sensing Image Recognition*. Vol 105. Elsevier; 2021:107252.
- Fout A, Byrd J, Shariat B, Ben-Hur A. Protein interface prediction using graph convolutional networks. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*; 2017:6533-6542.
- Goering S, Klein E. Fostering neuroethics integration with neuroscience in the BRAIN initiative: comments on the NIH neuroethics roadmap. *AJOB Neurosci.* 2020;11(3):184-188.
- Li X. Explain graph neural networks to understand weighted graph features in node classification. In: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer; 2020: 57-76.
- Jiang N, Jie W, Li J, Liu X, Jin D. GATrust: a multi-aspect graph attention network model for trust assessment in OSNs. *IEEE Trans Knowl Data Eng.* 2022.
- Chen C, Huang T. Camdar-adv: generating adversarial patches on 3D object. *Int J Intell Syst.* 2021;36(3): 1441-1453.
- Yan H, Chen M, Hu L, Jia C. *Secure Video Retrieval Using Image Query on an Untrusted Cloud*. Vol 97. Elsevier; 2020:106782.
- Sanchez-Gonzalez A, Heess N, Springenberg JT, et al. Graph networks as learnable physics engines for inference and control. In: *International Conference on Machine Learning*. PMLR; 2018:4470-4479.
- Battaglia P, Pascanu R, Lai M, Rezende DJ, Kavukcuoglu K. Interaction networks for learning about objects, relations and physics. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*; 2016:4509-4517.
- Tianqing Z, Zhou W, Ye D, Cheng Z, Li J. Resource allocation in IoT edge computing via concurrent federated reinforcement learning. *IEEE Internet Things J.* 2021;9(2):1414-1426.
- Meng W, Wang Y, Li W, Liu Z, Li J, Probst CW. Enhancing intelligent alarm reduction for distributed intrusion detection systems via edge computing. In: *Australasian Conference on Information Security and Privacy*. Springer; 2018:759-767.
- Yan H, Hu L, Xiang X, Liu Z, Yuan X. PPCL: privacy-preserving collaborative learning for mitigating indirect information leakage. *Inf Sci.* 2021;548:423-437.
- Li J, Ye H, Li T, et al. Efficient and secure outsourcing of differentially private data publishing with multiple evaluators. *IEEE Trans Dependable Secure Comput.* 2022;19(01):67-76.
- Hou S, Huang X, Liu JK, Li J, Xu L. Universal designated verifier transitive signatures for graph-based big data. *Inf Sci.* 2015;318:144-156.
- Chen B, Wang Z, Xiang T, Yang L, Yan H, Li J. ABAC: anonymous bilateral access control protocol with traceability for fog-assisted mobile crowdsensing. In: *International Conference on Data Mining and Big Data*. Springer; 2021:430-444.
- Ruiz L, Gama F, Ribeiro A. Gated graph recurrent neural networks. *IEEE Trans Signal Process.* 2020;68: 6303-6318.
- Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. *stat.* 2018;1050:4. <https://arxiv.org/abs/1710.10903>
- Tong Z, Liang Y, Sun C, Rosenblum DS, Lim A. Directed graph convolutional network. *arXiv preprint arXiv:2004.13970*; 2020.

21. Huang Z, Lin Z, Gong Z, Chen Y, Tang Y. A two-phase knowledge distillation model for graph convolutional network-based recommendation. *Int J Intell Syst.* 2022.
22. Chen X, Zhang F, Zhou F, Bonsangue M. Multi-scale graph capsule with influence attention for information cascades prediction. *Int J Intell Syst.* 2022;37(3):2584-2611.
23. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T. Collective classification in network data. *AI Mag.* 2008;29(3):93.
24. Namata G, London B, Getoor L, Huang B, Edu U. Query-driven active surveying for collective classification. In: *10th International Workshop on Mining and Learning with Graphs*; 2012:8.
25. McAuley J, Targett C, Shi Q, Van Den Hengel A. Image-based recommendations on styles and substitutes. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*; 2015:43-52.
26. Shchur O, Mumme M, Bojchevski A, Günnemann S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*; 2018.
27. García-Plaza AP, Fresno V, Unanue RM, Zubiaga A. Using fuzzy logic to leverage HTML markup for web page representation. *IEEE Trans Fuzzy Syst.* 2016;25(4):919-933.
28. Rozemberczki B, Allen C, Sarkar R. Multi-scale attributed node embedding. *J Complex Networks.* 2021;9(2):1-22.
29. Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY. A comprehensive survey on graph neural networks. *IEEE Trans Neural Networks Learn Syst.* 2020;32(1):4-24.
30. Hammond DK, Vandergheynst P, Gribonval R. Wavelets on graphs via spectral graph theory. *Appl Comput Harmonic Anal.* 2011;30(2):129-150.
31. Zhang J, Shi X, Xie J, Ma H, King I, Yeung DY. GaAN: gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*; 2018.
32. Cirstea RG, Guo C, Yang B. Graph attention recurrent neural networks for correlated time series forecasting-full version. *arXiv preprint arXiv:2103.10760*; 2021.
33. Zhao Y, Qi J, Liu Q, Zhang R. WGCN: graph convolutional networks with weighted structural features. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*; 2021:624-633.
34. Drummond C, Holte RC. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In: *Workshop on Learning from Imbalanced Datasets II*. Vol 11. Citeseer; 2003:1-8.
35. Shi M, Tang Y, Zhu X, Wilson D, Liu J. Multi-class imbalanced graph convolutional network learning. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*; 2020.
36. Rong Y, Huang W, Xu T, Huang J. DropEdge: towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*; 2019.
37. Chen D, Lin Y, Li W, Li P, Zhou J, Sun X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol 34; 2020:3438-3445.
38. Monti F, Boscaini D, Masci J, Rodola E, Svoboda J, Bronstein MM. Geometric deep learning on graphs and manifolds using mixture model CNNs. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2017:5115-5124.
39. Chapelle O, Scholkopf B, Zien A. Semi-supervised learning (Chapelle, O. et al., eds.; 2006)[book reviews]. *IEEE Trans Neural Networks.* 2009;20(3):542.
40. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*; 2016.
41. Haonan L, Huang SH, Ye T, Xiuyan G. Graph star net for generalized multi-task learning. *arXiv preprint arXiv:1906.12330*; 2019.
42. Gao Y, Yang H, Zhang P, Zhou C, Hu Y. GraphNAS: graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981*; 2019.
43. Zhang J, Zhang H, Xia C, Sun L. Graph-Bert: only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*; 2020.
44. Zhang J. Get rid of suspended animation problem: deep diffusive neural network on graph semi-supervised classification. *arXiv preprint arXiv:2001.07922*; 2020.

45. Rychalska B, Babel P, Gołuchowski K, Michałowski A, Dabrowski J, Biecek P. Cleora: a simple, strong and scalable graph embedding scheme. In: *International Conference on Neural Information Processing*. Springer; 2021:338-352.
46. Luo Y, Chen A, Yan K, Tian L. Distilling self-knowledge from contrastive links to classify graph nodes without passing messages. *arXiv preprint arXiv:2106.08541*; 2021.
47. Luo Y, Luo G, Yan K, Chen A. Inferring from references with differences for semi-supervised node classification on graphs. *Mathematics*. 2022;10(8):1262.
48. Pei H, Wei B, Chang KCC, Lei Y, Yang B. Geom-GCN: geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*; 2020.
49. Topping J, Di Giovanni F, Chamberlain BP, Dong X, Bronstein MM. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*; 2021.
50. Zhang F, Bu TM. CN-motifs perceptive graph neural networks. *IEEE Access*. 2021;9:151285-151293.
51. Yan Y, Hashemi M, Swersky K, Yang Y, Koutra D. Two sides of the same coin: heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462*; 2021.
52. Kulatililke GK, Portmann M, Ko R, Chandra SS. FDGATII: fast dynamic graph attention with initial residual and identity mapping. *arXiv preprint arXiv:2110.11464*; 2021.
53. Li X, Zhu R, Cheng Y, et al. Finding global homophily in graph neural networks when meeting heterophily. *arXiv preprint arXiv:2205.07308*; 2022.
54. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. *J Artif Intell Res*. 2002;16:321-357.
55. Japkowicz N, Stephen S. The class imbalance problem: a systematic study. *Intell Data Anal*. 2002;6(5):429-449.
56. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings*; 2010:249-256.
57. Clevert DA, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*; 2015.
58. Ren Y, Zhao P, Sheng Y, Yao D, Xu Z. Robust softmax regression for multi-class classification with self-paced learning. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*; 2017:2641-2647.
59. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res*. 2014;15(1):1929-1958.

How to cite this article: Pang Y, Huang T, Wang Z, et al. Graph Decipher: A transparent dual-attention graph neural network to understand the message-passing mechanism for the node classification. *Int J Intell Syst*. 2022;37:8747-8769. doi:10.1002/int.22966

APPENDIX A

A.1 | Data set

We experimentally validate our proposed algorithm on seven real-world graph data sets, as shown in Table A1, which summarizes the statistics of these seven data sets and the configuration of the train/val/test splits. The statistics summary of node attribute values in each data set is shown in Table A2.

A.1.1 | Overall

Cora and *Citeseer*²³ and *PubMed*²⁴: These three public data sets are graphs used to describe citation patterns of scientific publications. The nodes represent publications, while the edges

TABLE A1 Details of graph data sets in our experiments

Data set	#Classes	#Features	#Nodes	#Edges	#Train nodes	#Val nodes	#Test nodes
Cora	7	1433	2485	5069	140	500	1000
Citeseer	6	3703	2110	3668	120	500	1000
PubMed	3	500	19,717	44,324	60	500	1000
Coauthor CS	15	6805	18,333	81,894	300	500	1000
Amazon Photo	8	745	7487	119,043	160	500	1000
Coauthor Physics	5	8415	34,493	247,962	100	500	1000
Amazon Computers	10	767	13,381	245,778	200	500	1000

TABLE A2 Statistic of node attribute values in the above seven graph data sets

Data set	Feature values
Cora	[0, 1]
Citeseer	[0, 1]
PubMed	[0, 57 $\frac{1}{2}$]
Coauthor CS	[0 ~ 5]
Amazon Photo	[0, 1]
Coauthor Physics	[0 ~ 10, 14, 21, 28, 29, 37]
Amazon Computers	[0, 1]

Note: The symbol, 57 $\frac{1}{2}$, indicates the PubMed data set's 57 special fractions (feature values).

indicate the citation links among distinctive publications. For the Cora and Citeseer data sets, dictionaries (feature vectors) are utilized to explore the most common words that appear in these publications. Thus, each publication is described by a 0/1 value, which indicates the absence/existence of the corresponding word from the dictionary. While for the publications in the PubMed data set, a term frequency-inverse document frequency (TF/IDF) is used to calculate the separation between them. In summary, the Cora data set includes 2708 nodes with 5429 links in seven categories, and the dimension of each node feature is 1433. The Citeseer data set consists of 3327 nodes with 4732 links in six categories, and each node feature has 3703 dimensions. The PubMed data set contains 19,717 nodes with 44,338 links in three categories and 500 dimensions per node feature vector.

*Amazon Computers and Amazon Photo*²⁵: These two data sets represent two different Amazon copurchase graphs. Each node denotes products in different categories, while edges show two interests in bundle sales. And each dimension of the node features represents bag-of-words encoded product reviews. In summary, the Amazon Computers data set includes 11,381 nodes with 245,778 links in 10 categories, and the dimension of each node feature is 767. On the other hand, the Amazon Photo data set consists of 7487 nodes with 119,043 links in eight categories, and each node feature has 745 dimensions.

*Coauthor CS and Coauthor Physics*²⁶: In both coauthorship graphs, each node represents an author, while the edge indicates two nodes coauthored a paper. The authors are grouped into different active fields or categories. Node feature vector illustrates the paper's keywords for each node, which represents the author's article. In summary, the Coauthor CS data set includes 18,333 nodes with 81,894 links in 15 categories, and the dimension of each node feature is 6805. The Coauthor Physics data set consists of 34,493 nodes with 247,962 links in five categories, and each node feature has 8415 dimensions.

A.1.2 | Distribution of node categories

This section summarizes the distribution of the node categories of all seven data sets, as illustrated in Table A3.

Cora: 2708 nodes in seven categories. Cora data set holds the second least number of nodes among all seven data sets. The number of the top node class (#4) is 818, while the smallest node class (#7) only exists 180. The ratio between the majority and minority is around 9:2.

Citeseer: 2110 nodes in six categories. The Citeseer data set has the least number of nodes among all seven graph data sets. The number of the top node class (#4) is 701, while the smallest node class (#1) only exists 249. The ratio of the majority and minority is around 3:1.

PubMed: 19,717 nodes in three categories. PubMed data set contains the least types of the node category among all seven graph data sets. The number of the top node class (#3) is 7875, while the smallest node class (#7) only exists 4103. The ratio of the majority and minority is around 2:1.

Coauthor CS: 18,333 nodes in 15 categories. Coauthor CS data set has the most types of the node category among all seven graph data sets. The number of the top node class (#14) is 4136, while the smallest node class (#10) only exists 118. The ratio of the majority and minority is around 35:1. Thus, the distribution of this graph data set is the most imbalanced in our experiment.

Coauthor Physics: 34,493 nodes in five categories. Coauthor Physics data set obtains the most nodes among all seven graph data sets. The number of the top node class (#3) is 17,426, while the smallest node class (#10) only exists 2753. The ratio of the majority and minority is around 6:1.

TABLE A3 Distributions of node categories in different graph data sets are illustrated

Data set	#1	#2	#3	#4	#5	#6	#7	#8
Cora	351	217	418	818	426	298	180	–
Citeseer	249	590	668	701	596	508	–	–
PubMed	4103	7739	7875	–	–	–	–	–
Coauthor CS [§]	708	462	2050	429	1394	2193	371	924
Coauthor CS [#]	775	118	1444	2033	420	4136	876	–
Amazon Photo	369	1686	703	915	882	823	1941	331
Coauthor Physics	5750	5045	17,426	2753	3519	–	–	–
Amazon Computers*	436	2142	1414	542	5158	308	487	818

Note: The symbols § and # represent the first eight and the following seven categories of the Coauthor CS data set. The third symbol * indicates the first eight categories in the Amazon Computers data set, containing another two categories with 2156 and 291 nodes separately.

Amazon Photo: 7487 nodes in eight categories. The number of the top node class (#7) is 1941, while the smallest node class (#8) only exists 331. The ratio of the majority and minority is around 6:1.

Amazon Computers: 13,381 nodes in 10 categories. Amazon Computers data set involves the second largest number of node categories among all seven graph data sets. The number of the top node class (#5) is 5158, while the smallest node class (#10) only exists 291. The ratio of the majority and minority is around 18:1.

A.2 | Experimental setup

To avoid the gradient from exploding or vanishing during the learning process, we chose Glorot and Bengio⁵⁶ to initialize the parameters of GD. The exponential linear unit⁵⁷ yields nonlinear outputs at the end of both modules, NAB and FAB. The softmax⁵⁸ is used to send the probability distribution over predicted node categories at the end of GD. Moreover, during the training process, the dropout approach⁵⁹ is introduced to avoid overfitting, and the dropout rate is set to a range of 0.3–0.7 depending on the data set. The GFP size, s , is set as 2, and the number of multiheads, ζ , is applied to 8.

A.3 | Ablation study

In this section, a comprehensive analysis of our network is provided. Section A.3.1 demonstrates the impact of GFP size for the network performance and computation, and Section A.3.2 discusses the effectiveness of multiheads architecture based on the proposed single-head layer.

A.3.1 | Filter size of the GFP

In GFP, the graph feature filter's size is a significant parameter that affects GD's performance and computation burden as it determines the amount of local dominant and representative features by category in the learning process. The GD's performance as a function of filter size applied to our seven data sets is shown in Figure A1. In these tests, the accuracy and number of parameters are indicators of a network's performance and computation burden.

As shown in Figure A1, the GD with feature pooling size 2 (GD-2) performs nearly as well as or better than most algorithms tested. The GD-2 algorithm typically imposes a slightly more significant computation burden than competing algorithms, though the gains in performance are clear. By computing only representative attributes in the GFP filter, the GD-2 algorithm can perform more efficiently. We found the optimal GFP size to be 2, effectively balancing the network's performance and computation burden. Filter size of 3 (GD-3) offers much lower computation burdens; however, this comes at the expense of the network performance, as shown in each test. These experiments show that the GFP module successfully preserves the representative attributes under the node classification task, achieving clear performance gains with greater network capacity.

A.3.2 | Multiheads architecture

This section demonstrates how the addition of heads impacts the overall performance of the network. We evaluated the performance and complexity of GD on all seven graph data sets as a function of heads under the node classification task, which are illustrated in Tables A4 and A5. The floating-point operations per second (FLOPs) and average precision (AP) are network complexity and performance indicators. As heads were added, an improvement trend was observed to the point of diminishing returns, usually when around 8 or 10 heads are employed.

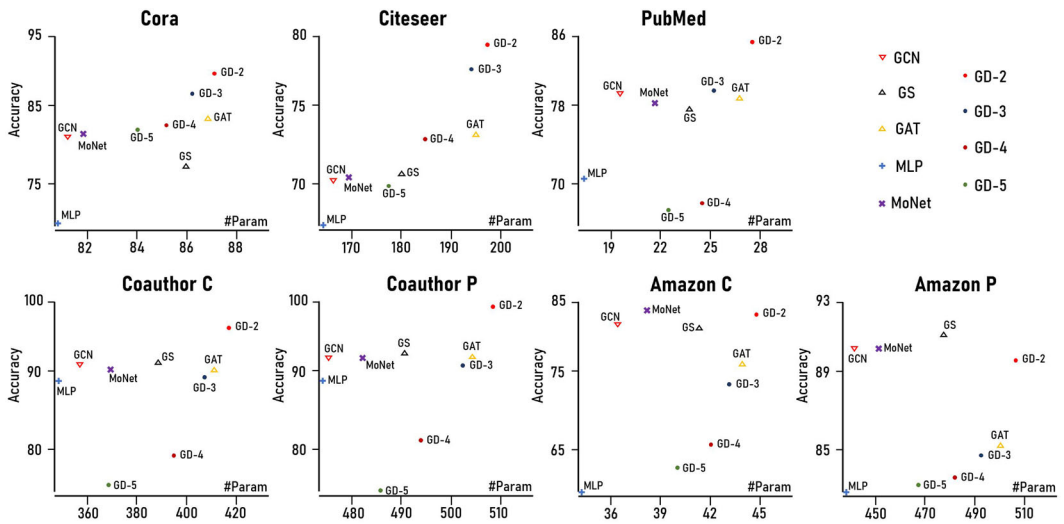


FIGURE A1 Performance (accuracy%) and parameters of different networks on seven data sets under the node classification task. GAT, graph attention network; GCN, graph convolutional network; GD-x, Graph Decipher with a graph filter size of x; GS, GraphSAGE; MLP, multilayer perceptron. [Color figure can be viewed at wileyonlinelibrary.com]

TABLE A4 Balance between complexity (MFLOPs) and accuracy of our network with distinctive multiheads on Cora, Citeseer, and PubMed data sets

Heads	Cora		Heads	Citeseer		Heads	PubMed	
	Flops (M)	AP		Flops (M)	AP		Flops (M)	AP
2	22.01	85.7	2	27.03	76.6	2	1166.64	83.3
4	36.67	86.3	4	45.01	77.3	4	1945.18	83.9
6	51.33	87.3	6	63.01	77.9	6	2719.16	85.1
8	66.01	88.3	8	81.01	78.9	8	3498.94	85.5
10	80.67	88.5	10	99.05	79.0	10	4277.67	85.7

Note: Bold values indicate the optimal choice by trading off the complexity and average precision.

Abbreviation: AP, average precision.

On the Cora, Citeseer, PubMed, and Coauthor data sets, dual-head performance of GD is 85.7%, 76.6%, 83.3%, 93.7%, and 94.9%, respectively, while the GAT achieves 83.4%, 72.5%, 79.0%, 90.5%, and 90.5%. These experiments demonstrate GD's superior ability to push the upper limit of an existing network's performance. When using four and six parallel heads, the performance trend continues to improve dramatically. The eight-head configuration improves over the six-head configuration by 1.0% and costs 14.08M FLOPs on the Cora data set. Beyond an eight-head configuration, the trend begins to subside, as there is only a marginal 0.2% gain in performance at 10-heads while costing a substantial 14.66M FLOPs. These results indicate that the

TABLE A5 Balance between complexity (MFLOPs) and accuracy of our network with distinctive multiheads on Coauthor CS, Coauthor Physics, Amazon Computer, and Amazon Photo data sets

Heads	Coauthor C		Heads	Coauthor P		Heads	Amazon C		Heads	Amazon P	
	Flops (M)	AP		Flops (M)	AP		Flops (M)	AP		Flops (M)	AP
2	1008.62	93.7	2	3568.26	94.9	2	537.07	81.3	2	168.21	87.1
4	1681.04	94.5	4	5982.16	95.8	4	895.12	81.9	4	281.36	87.9
6	2353.46	95.0	6	8326.02	96.6	6	1248.17	82.8	6	389.53	89.0
8	3025.87	95.9	8	10,709.76	97.3	8	1611.41	83.1	8	526.70	89.2
10	3698.29	95.9	10	13,083.77	97.5	10	1969.26	83.2	10	616.88	89.2

Note: Bold values indicate the optimal choice by trading off the complexity and average precision.

Abbreviation: AP, average precision.

eight-head configuration of GD achieves high performance with an optimally balanced complexity on the Cora, Citeseer, and Coauthor data sets, and on the PubMed and Amazon data sets, the six-head configuration of GD is the ideal option considering the trade-offs in performance and complexity.