GIPA: A General Information Propagation Algorithm for Graph Learning

Houyi Li^{1*}, Zhihong Chen², Zhao Li^{3,6}, Qinkai Zheng⁴, Peng Zhang⁵, and Shuigeng Zhou^{1**}

- $^{1}\,$ School of computer science, Fudan University, Shanghai, China
 - ² Alibaba Group, Hangzhou, China
 - ³ Zhejiang University, Hangzhou, China ⁴ Tsinghua University, Beijing, China
- Cyberspace Institute of Advanced Technology, Guangzhou University, China Link2Do Technology, Hangzhou, China

lihouyi2008@126.com, jhon.czh@alibaba-inc.com, lzjoey@gmail.com, qinkai.zheng1028@gmail.com, p.zhang@gzhu.edu.cn, sgzhou@fudan.edu.cn

Abstract. Graph neural networks (GNNs) have been widely used in graph-structured data computation, showing promising performance in various applications such as node classification, link prediction, and network recommendation. Existing works mainly focus on node-wise correlation when doing weighted aggregation of neighboring nodes based on attention, such as dot product by the dense vectors of two nodes. This may cause conflicting noise in nodes to be propagated when doing information propagation. To solve this problem, we propose a General Information Propagation Algorithm (GIPA), which exploits more fine-grained information fusion including bit-wise and feature-wise correlations based on edge features in their propagation. Specifically, the bit-wise correlation calculates the element-wise attention weights through a multi-layer perceptron (MLP) based on the dense representations of two nodes and their edge: The feature-wise correlation is based on the one-hot representations of node attribute features for feature selection. We evaluate the performance of GIPA on the Open Graph Benchmark proteins (OGBNproteins) dataset and the Alipay dataset of Alibaba Group. Experimental results reveal that GIPA outperforms the state-of-the-art models in terms of prediction accuracy, e.g., GIPA achieves an average ROC-AUC of 0.8917 ± 0.0007 , which is better than that of all the existing methods listed in the OGBN-proteins leaderboard.

Keywords: Graph neural networks, Fine-grained information fusion, Bit-wise and feature-wise attention.

1 Introduction

Graph representation learning typically aims to learn an informative embedding for each graph node based on the graph topology (link) information. Generally,

^{*} Contributed to this work when the author worked in Ant and Alibaba Group.

^{**} Shuigeng Zhou is the corresponding author.

the embedding of a node is represented as a low-dimensional feature vector, which can be used to facilitate downstream applications. This research focuses on homogeneous graphs that have only one type of nodes and one type of edges. The purpose is to learn node representations from the graph topology [9,28,7]. Specifically, given a node u, either breadth-first search, depth-first search or random walk is used to identify a set of neighboring nodes. Then, u's embedding is learnt by maximizing the co-occurrence probability of u and its neighbors. Early studies on graph embedding have limited capability to capture neighboring information from a graph because they are based on shallow learning models such as SkipGram [24]. Moreover, transductive learning is used in these graph embedding methods, which cannot be generalized to new nodes that are absent in the training graph.

Graph neural networks [15,10,34] are proposed to overcome the limitations of traditional graph embedding models. GNNs employ deep neural networks to aggregate feature information from neighboring nodes and thereby have the potential to gain better aggregated embeddings. GNNs can support inductive learning and infer the class labels of unseen nodes during prediction [10,34]. The success of GNNs is mainly due to the neighborhood information aggregation. However, GNNs face two challenges: which neighboring nodes of a target node are involved in message passing? and how much contribution each neighboring node makes to the aggregated embedding? For the former question, neighborhood sampling [10,42,4,13,46,14] is proposed for large dense or power-law graphs. For the latter, neighbor importance estimation is used to attach different weights to different neighboring nodes during feature propagation. Importance sampling [4,46,14] and attention [34,21,37,43,12] are two popular techniques. Importance sampling is a special case of neighborhood sampling, where the importance weight of a neighboring node is drawn from a distribution over nodes. This distribution can be derived from normalized Laplacian matrices [4,46] or jointly learned with GNNs [14]. With this distribution, at each step a subset of neighbors is sampled, and aggregated with the importance weights. Similar to importance sampling, attention also attaches importance weights to neighbors. Nevertheless, attention differs from importance sampling. Attention is represented as a neural network and is always learned as a part of a GNN model. In contrast, importance sampling algorithms use statistical models without trainable parameters.

Existing attention mechanisms consider only the correlation of node-wise, ignoring the suppression of noise information in transmission, and the information of edge features. In real world applications, only partial users authorize the system to collect theirs profiles. The model cannot learn the node-wise correlation between a profiled user and a user we know nothing about. Therefore, existing models will spread noise information, resulting in inaccurate node representations. However, two users who often transfer money to each other and two users who only have a few conversations have different correlation.

In this paper, to solve the problems mentioned above, we present a new graph neural network attention model, namely General Information Propagation Algorithm (GIPA). We design a bit-wise correlation module and a feature-wise correlation module. Specifically, we believe that each dimension of the dense vector represents a feature of the node. Therefore, the bit-wise correlation module filters at the dense representation level. The dimension of attention weights is equal to that of density vector. In addition, we represent each attribute feature of the node as a one-hot vector. The feature-wise correlation module performs feature selection by outputting the attention weights of similar dimensionality and attribute features. It is worth mentioning that to enable the model to extract better attention weights, edge features that measure the correlation between nodes are also included in the calculation of attention. Finally, GIPA inputs sparse embedding and dense embedding into the wide end and deep end of the deep neural network for learning specific tasks, respectively. Our contributions are summarized as follows:

- 1) We design the bit-wise correlation module and the feature-wise correlation module to perform more refined information weighted aggregation from the element level and the feature level, and utilize the edge information.
- 2) Based on the wide & deep architecture [6], we use dense feature representation and sparse feature representation to extract deep information and retain shallow original information respectively, which provides more comprehensive information for the downstream tasks.
- 3) Experiments on the Open Graph Benchmark (OGB) [11] proteins dataset (OGBN-proteins) demonstrate that GIPA achieves better accuracy with an average ROC-AUC of 0.8917 ± 0.0007 ⁷ than the state-of-the-art methods listed in the OGBN-proteins leaderboard ⁸. In addition, GIPA has been tested on billion-scale industrial Alipay dataset.

2 Related Work

In this section, we review existing attention primitive implementations in brief. [2] proposes an additive attention that calculates the attention alignment score using a simple feed-forward neural network with only one hidden layer. The alignment score score(q, k) between two vectors q and k is defined as $score(q, k) = u^T \tanh(W[q; k])$, where u is an attention vector and the attention weight $\alpha_{q,k}$ is computed by normalizing $scor\tilde{e}_{q,k}$ over all q,k values with the softmax function. The core of the additive attention lies in the use of attention vector u. This idea has been widely adopted by several algorithms [41,27] for natural language processing. [23] introduces a global attention and a local attention. In global attention, the alignment score can be computed by three alternatives: dot-product $(q^T k)$, general $(q^T W k)$ and concat (W[q; k]). In contrast, local attention computes the alignment score solely from a vector (Wq). Likewise, both global and local attention normalize the alignment scores with the softmax function. [33] proposes a self-attention mechanism based on scaled dot-product. This

 $^{^7}$ The reproducible code is open source: $\label{eq:https://github.com/houyili/gipa_wide_deep}$

⁸ https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-proteins

self-attention computes the alignment score between any q and k as follows: $scor\tilde{e}_{q,k} = q^T k/\sqrt{d_k}$. This attention differs from the dot-product attention [23] by only a scaling factor of $\frac{1}{\sqrt{d_k}}$. The scaling factor is used because the authors of [33] suspected that for large values of d_k , dot-product is large in magnitude, which thereby pushes the softmax function into regions where it has extremely small gradients. More attention mechanisms, like feature-wise attention [30] and multi-head attention [33], can be referred to some surveys [3,25].

In addition, these aforementioned attention primitives have been extended to heterogeneous graphs. HAN [37] uses a two-level hierarchical attention that consists of a node-level attention and a semantic-level attention. In HAN, the node-level attention learns the importance of a node to any other node in a meta-path, while the semantic-level one weighs all meta-paths. HGT [44] weakens the dependency on meta-paths and instead uses meta-relation triplets as basic units. HGT uses node-type-aware feature transformation functions and edge-type-aware multi-head attention to compute the importance of each edge to a target node. It is worthy of mentioning that heterogeneous models must not always be superior to homogeneous ones, and vice versa. In this paper, unlike the node-wise attention in existing methods that ignores noise propagation, our proposed GIPA introduces two MLP-based correlation modules, the bit-wise correlation module and the feature-wise correlation module, to achieve fine-grained selective information propagation and utilize the edge information.

3 Methodology

3.1 Preliminaries

Graph Neural Networks. Consider an attributed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where \mathcal{V} is the set of nodes and \mathcal{E} is the set of edges. GNNs use the same model framework as follows [10,39]:

$$\tilde{h}_i^k = \phi(\tilde{h}_i^{k-1}, \mathcal{F}_{agg}(\{\tilde{h}_j^{k-1}, j \in \mathcal{N}(i)\}))$$
(1)

and where \mathcal{F}_{agg} represents an aggregation function, ϕ represents an update function, \tilde{h}_i^k represents the node embedding based on node feature x_i^k , and $\mathcal{N}(i)$ represents the neighbor node set. The objective of GNNs is to update the embedding of each node by aggregating the information from its neighbor nodes and the connections between them.

3.2 Model Architecture

In this section, we present the architecture of GIPA in Fig. 1, which extracts node features and edge features in a more general way. Consider a node i with feature embedding \tilde{h}_i and its neighbor nodes $j \in \mathcal{N}(i)$ with feature embedding \tilde{h}_j . $\tilde{e}_{i,j}$ represents the edge feature between node i and j. The problem is how to generate an expected embedding for node i from its own node feature embedding \tilde{h}_i , its neighbors' node features embedding $\{\tilde{h}_j\}$ and the related edge features

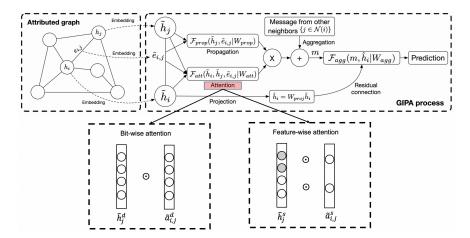


Fig. 1: The architecture of GIPA, which consists of attention, propagation, and aggregation modules (or processes). The red shadow indicates the bit-wise module and feature-wise module, which extract neighbor information more relevant to the current node through more fine-grained selective information fusion.

 $\{\tilde{e}_{i,j}\}$. The workflow of GIPA consists of three major modules (or processes): attention, propagation and aggregation. First, GIPA computes the dense embedding \tilde{h}_i^d , \tilde{h}_j^d , and $\tilde{e}_{i,j}$, and sparse embedding \tilde{h}_i^s , \tilde{h}_j^s from raw features. Then, the attention process calculates the bit-wise attention weights by using fully-connected layers on \tilde{h}_i^d , \tilde{h}_j^d , and $\tilde{e}_{i,j}$ and feature-wise by using fully-connected layers on \tilde{h}_i^s , \tilde{h}_j^s , and $\tilde{e}_{i,j}$. Following that, the propagation process focuses on propagating information of each neighbor node j by combining j's node embedding \tilde{h}_j with the associated edge embedding $\tilde{e}_{i,j}$. Finally, the aggregation process aggregates all messages from neighbors to update the embedding of i. The following subsections introduce the details of each process in one layer.

Embedding Layer. The GIPA (wide & deep) is more suitable for these scenarios: each node in GNN is not composed of text or images, but represents objects such as users, products, and proteins, whose features are composed of category features and statistical features. For example, the category feature in *ogbn-proteins* dataset is what species the proteins come from. And the statistical features in *Alipay* dataset are similar to the total consumption of users in a year.

For dense embedding, each integer number can express a category, each floating-point number can express a statistical value, thus a one-dimensional embedding can represent one feature. However, the sparse embedding of one feature requires more dimensions. For each category feature, one-hot encoding is required. For example, a certain "category feature" has a total of K possible categories, and an K+1 dimensional vector is required to represent the feature. And each "statistical feature" is cut into K categories (using equal-frequency or equal-width cutting method), and then K+1 dimensional one-hot encoding

Algorithm 1 The optimization strategy of GIPA

```
1: Input: Graph \mathcal{G} = \{\mathcal{V}, \mathcal{E}\}; input features \{x_v, \forall v \in V\}; Number of layer K
  2: Note: \tilde{h}_v^* \in \{\tilde{h}_v^s, \tilde{h}_v^d\}
  3: \tilde{h}_v^{d^0} \leftarrow DenseEmb(x_v), \forall v \in V
  4: \tilde{h}_{v}^{s_0} \leftarrow SparseEmb(x_v), \forall v \in V
          while not end of epoch do
                        Select a subgraph \mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\} \in \mathcal{G}
  6:
                        for each k \in [1, K] do
  7:
                                   if k > 1
\tilde{h}_{i}^{*k-1} \leftarrow o_{i}^{*k-1}, \forall v_{i} \in V_{t}
\tilde{a}_{i,j}^{*k} \leftarrow \mathcal{F}_{act}(\mathcal{F}_{att}^{*k}(\tilde{h}_{i}^{*k-1}, \tilde{h}_{j}^{*k-1}, \tilde{e}_{i,j}|W_{att}^{*})), \forall v_{i} \in V_{t}
m_{i,j}^{*k} \leftarrow \tilde{a}_{i,j}^{*k} * \mathcal{F}_{prop}^{*}(\tilde{h}_{j}^{*k-1}, \tilde{e}_{i,j}|W_{prop}^{*}), \forall v_{i} \in V_{t}
o_{i}^{*k} \leftarrow \mathcal{F}_{agg}^{*}(\sum_{j \in \mathcal{N}(i)} m_{i,j}^{*k}, \hat{h}_{i}^{*k-1}|W_{agg}^{*}), \forall v_{i} \in V_{t}
  8:
  9:
10:
11:
12:
                         \hat{y}_i = Deep(o_i^{dk}) + W_{wide}(o_i^{sk})
13:
                        L \leftarrow \sum_{\forall v_i \in V_t} \hat{l}(y_i, \hat{y}_i)
Update all parameters by gradient of L
14:
15:
```

is performed. Thus, the concatenations of dense and sparse embeddings are the inputs of deep part and wide part respectively.

Attention Process. Different from the existing attention mechanisms like self-attention or scaled dot-product attention, we use MLP to realize a bit-wise attention mechanism and a feature-wise attention mechanism. The bit-wise and feature-wise *attention* process of GIPA can be formulated as follows:

$$\boldsymbol{a}_{i,j}^{d} = \mathcal{F}_{att}^{d}(\tilde{h}_{i}^{d}, \tilde{h}_{j}^{d}, \tilde{e}_{i,j} | W_{att}^{d}) = \text{MLP}([\tilde{h}_{i}^{d} | | \tilde{h}_{j}^{d} | | \tilde{e}_{i,j}] | W_{att}^{d})$$
 (2)

$$\boldsymbol{a}_{i,j}^{s} = \mathcal{F}_{att}^{s}(\tilde{h}_{i}^{s}, \tilde{h}_{j}^{s}, \tilde{e}_{i,j}|W_{att}^{s}) = \text{MLP}([\tilde{h}_{i}^{s}||\tilde{h}_{j}^{s}||\tilde{e}_{i,j}]|W_{att}^{s})$$
(3)

where $\mathbf{a}_{i,j}^d \in \mathcal{R}^n$ is bit-wise attention weight and its dimension is the same as that of \tilde{h}_i^d , $\mathbf{a}_{i,j}^s \in \mathcal{R}^m$ is feature-wise attention weight and its dimension is the same as the number of node features, the attention function \mathcal{F}_{att}^* is realized by an MLP with learnable weights W_{att}^* (without bias). Its input is the concatenation of the node embeddings \tilde{h}_i^* and h_j^* as well as the edge embedding $\tilde{e}_{i,j}$. As the edge features $\tilde{e}_{i,j}$ measuring the correlation between nodes are input into MLP, this attention mechanism could be more representative than previous ones simply based on dot-product. The final attention weight is calculated by an activation function for bit-wise part and feature-wise part:

$$\tilde{\boldsymbol{a}}_{i,j}^* = \mathcal{F}_{act}(\boldsymbol{a}_{i,j}^*) \tag{4}$$

where \mathcal{F}_{act} represents the activation function, such as *softmax*, *leaky-relu*, *soft-plus*, etc. Based on the experimental results, we finally define the activation function as *softplus*. Details can be seen in Section 4.3.

Propagation Process. Unlike GAT [34] that considers only the node feature of neighbors, GIPA incorporates both node and edge embeddings during the *propagation* process:

$$p_{i,j}^d = \mathcal{F}_{prop}^d(\tilde{h}_i^d, \tilde{e}_{i,j}|W_{prop}^d) = \text{MLP}([\tilde{h}_i^d||\tilde{e}_{i,j}]|W_{prop}^d), \tag{5}$$

$$p_{i,j}^s = \mathcal{F}_{prop}^s(\tilde{h}_j^s, \tilde{e}_{i,j}|W_{prop}^s) = \text{MLP}([\tilde{h}_j^s||\tilde{e}_{i,j}]|W_{prop}^s), \tag{6}$$

where the propagation function \mathcal{F}_{prop}^* is also realized by an MLP with learnable weights W_{prop}^* . Its input is the concatenation of a neighbor node dense and sparse embeddings \tilde{h}_j^* and the related edge embedding $\tilde{e}_{i,j}$. Thus, the *propagation* is done bit-wise and feature-wise rather than node-wise.

Combining the results by attention and propagation by bit-wise and featurewise multiplication, GIPA gets the message $m_{i,j}^d$ and $m_{i,j}^s$ of node i from j:

$$m_{i,j}^d = \tilde{a}_{i,j}^d * p_{i,j}^d \quad m_{i,j}^s = \tilde{a}_{i,j}^s \otimes p_{i,j}^s$$
 (7)

Aggregation Process. For each node i, GIPA repeats previous processes to get messages from its neighbors. The *aggregation* process first gathers all these messages by a reduce function, summation for example:

$$m_i^* = \sum_{j \in \mathcal{N}(i)} m_{i,j}^* \tag{8}$$

Then, a residual connection between the linear projection \hat{h}_i^* and the message of m_i is added through concatenation:

$$\hat{h}_i^d = W_{proj}^d \tilde{h}_i^d \quad \hat{h}_i^s = W_{proj}^s \tilde{h}_i^s \tag{9}$$

$$o_i^d = \mathcal{F}_{agg}^d(m_i^d, \tilde{h}_i^d | W_{agg}^d) = \text{MLP}([m_i^d | | \tilde{h}_i^d] | W_{agg}^d) \oplus \hat{h}_i^d$$
 (10)

$$o_i^s = \mathcal{F}_{agg}^s(m_i^s, \tilde{h}_i^s | W_{agg}^s) = \text{MLP}([m_i^s | | \tilde{h}_i^s] | W_{agg}^s) \oplus \hat{h}_i^s$$
(11)

where an MLP with learnable weights W^*_{agg} is applied to get the final dense output o^d_i and sparse output o^s_i . Finally, we would like to emphasize that the process of GIPA can be easily extended to multi-layer variants by stacking the process multiple times. After we get the aggregated output of the node, o^d_i and o^s_i respectively input the depth side and wide side of the Deep&Wide architecture for downstream tasks. See algorithm 1 for details.

4 Experiments

4.1 Datasets and Settings

Datasets. In our experiments, we choose two edge-attribute dataset: the ogbn-proteins dataset from OGB [11] and Alipay dataset [18]. The ogbn-proteins dataset is an undirected and weighted graph, containing 132,534 nodes of 8 different species and 79,122,504 edges with 8-dimensional features. The task is a multilabel binary classification problem with 112 classes representing different protein functions. The Alipay dataset is an edge attributed graph, containing 1.40 billion nodes with 575 features and 4.14 billion edges with 57 features. The task is a multi-label binary classification problem. It is worth noting that due to the high cost of training on Alipay data set, we only conduct ablation experiments on the input features of \mathcal{F}_{att} and \mathcal{F}_{prop} in the industrial data set, as Table 3.

Baselines. Several representative GNNs including SOTA GNNs are used as baselines. For semi-supervised node classification, we utilize GCN [15], Graph-SAGE [10], GAT [34], MixHop [1], JKNet [40], DeeperGCN [17], GCNII [5], DAGNN [20], MAGNA [35], UniMP [31], GAT+BoT [38], RevGNN [16] and AGDN [32]. Note that DeeperGCN, UniMP, RevGNN, and AGDN are implemented with random partition. GAT is implemented with neighbor sampling. Except for our GIPA, results of other methods are from their papers or the OGB leaderboard.

Evaluation metric. The performance is measured by the average ROC-AUC scores. We follow the dataset splitting settings as recommended in OGB and test the performance of 10 different trained models with different random seeds. **Hyperparameters.** For the number of layers K, we search the best value from 1 to 6. As for the activation function of attention process, we consider common activation functions. For details, please refer to Section 4.3.

Running environment. For ogbn-proteins dataset, GIPA is implemented in Deep Graph Library (DGL) [36] with Pytorch [26] as the backend. Experiments are done in a platform with Tesla V100 (32G RAM). For Alipay dataset, GIPA is implemented in *GraphTheta* [18], and runs on private cloud of Alibaba Group.

4.2 Performance Comparison

Table 1 shows the average ROC-AUC and the standard deviation for the test and validation set. The results of the baselines are retrieved from the ogbn-proteins leaderboard⁸. Our GIPA outperforms all previous methods in the leaderboard and reaches an average ROC-AUC higher than 0.89 for the first time. Furthermore, GIPA only with 3 layer achieved the *SOTA* performance on *ogbn-proteins* dataset. This result shows the effectiveness of our proposed bit-wise and featurewise correlation modules, which can leverage the edge features to improve the performance by fine-grained information fusion and noise suppression.

To further investigate the impact of each component in our proposed GIPA, we conduct the ablation study on the ogbn-proteins dataset. As shown in Table 2, compare with GIPA w/o bit-wise module, GIPA w/o feature-wise module. And

Table 1: Test and validation performance results (ROC-AUC) on the *ogbn-proteins* dataset. The improvements over comparison methods are statistically significant at 0.05 level.

Method	Test ROC-AUC	Validation ROC-AUC
GCN	0.7251 ± 0.0035	0.7921 ± 0.0018
GraphSAGE	0.7768 ± 0.0020	0.8334 ± 0.0013
DeeperGCN	0.8580 ± 0.0028	0.9106 ± 0.0011
GAT	0.8682 ± 0.0018	0.9194 ± 0.0003
UniMP	0.8642 ± 0.0008	0.9175 ± 0.0006
GAT+BoT	0.8765 ± 0.0008	0.9280 ± 0.0008
RevGNN-deep	0.8774 ± 0.0013	0.9326 ± 0.0006
RevGNN-wide	0.8824 ± 0.0015	0.9450 ± 0.0008
AGDN	0.8865 ± 0.0013	0.9418 ± 0.0005
GIPA-3Layer	0.8877 ± 0.0011	0.9415 ± 0.0023
GIPA-6Layer	$\bf 0.8917 \pm 0.0007$	$\bf 0.9472 \pm 0.0020$

Table 2: Ablation study on the ogbn-proteins dataset.

Method	Test ROC-AUC	Validation ROC-AUC
GIPA w/o bit-wise module	0.8813 ± 0.0011	0.9332 ± 0.0009
GIPA w/o feature-wise module	0.8701 ± 0.0021	0.9320 ± 0.0007
GIPA w/o edge feature	0.8599 ± 0.0047	0.9204 ± 0.0038
GIPA	0.8917 ± 0.0007	0.9472 ± 0.0020

the combination of these components (i.e., GIPA) yields the best performance, which indicates the necessity of bit-wise and feature-wise correlation modules.

The average training times per epoch on *ogbn-proteins* with GIPA, 3-Layer GIPA, AGDN are 8.2 seconds (s), 5.9s, and 4.9s, respectively. The average inference times on whole graph of *ogbn-proteins* with GIPA, 3-Layer GIPA, AGDN are 11.1s, 9.3s, and 10.7s, respectively. Compared with ADGN, GIPA is slower in training speed, but has advantages in inference speed.

4.3 Hyperparameter Analysis and Ablation Study

Effect of the number of layers K. To study the impact of the number of layers K on performance, we vary its value to $\{1, 2, 3, 4, 5, 6\}$. As shown in Fig. 2a, with the increase of layers, the performance of the model on the test set gradually converges. However, in the comparison between the five layers and the six layers, the increase in the performance of the model on the verification set is far greater than that on the test set. This result shows that with the increase of complexity, the performance of the model can be improved, but there will be an over-fitting problem at a bottleneck. Therefore, we finally set K to 6.

Analysis on Attention Process with Correlation Module. Because the bit-wise correlation module and the feature-wise correlation module are the most important parts of GIPA, we make a detailed analysis of their architecture, including activation function and edge feature.

Table	e 3 : Ab	lation	on	Апрау	aatas	et. 'S	and	.'D'	are j	primary	ana	neighbo	r noaes	٠.

\mathcal{F}_{att} input	\mathcal{F}_{prop} input	Feature-wise module	ROC-AUC	F1
Edge	D. Node	None	0.8784	0.1399
Edge & D. Node	Edge & D. Node	None	0.8916	0.1571
Edge & D.+ S. Nodes	s Edge & D. Node	None	0.8943	0.1611
Edge & D.+ S. Nodes	s Edge & D. Node	Yes	0.8961	0.1623

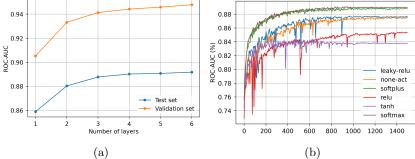


Fig. 2: Effect of hyperparameters. (a) ROC-AUC vs. the number of layers K. (b) Convergence on test set vs. activation function of attention.

1) Activation function: as shown in Fig. 2b, by comparing these activation functions: the performance of tanh and relu are even worse than the one without activator. The model using softmax as the activation function achieves the best performance, but the performance of model using softplus is close to it. This phenomenon indicates that the attention module of GIPA can adaptively learn the correlation weight between nodes. Cause $\sum_{j \in N(i)} a_{ij}$ is needed to be calculated firstly by aggregation from all neighbors, the normalization operation of softmax is a costly operation in GNN. On the other hand, GIPA with softplus trains faster than that with softmax on ogbn-proteins. Considering the trade-off between cost and performance, softplus is adopted on billion-scale Alipay dataset. 2) Edge feature: As shown in Table 2, we find that the performance of GIPA is better than that of GIPA w/o edge feature, which means that the edge feature contains the correlation information between two nodes. Therefore, it is an indispensable feature in attention processing and can help the two correlation modules get more realistic attention weights.

5 Conclusion

We have presented GIPA, a new graph attention network architecture for graph data learning. GIPA consists of a bit-wise correlation module and a feature-wise correlation module, to leverage edge information and realize the fine granularity information propagation and noise filtering. Performance evaluation on the ogbn-proteins dataset has shown that our method outperforms the state-of-the-art methods listed in the ogbn-proteins leaderboard. And it has been tested on the billion-scale industrial dataset.

References

- 1. Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyun-yan, H., Ver Steeg, G., Galstyan, A.: Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In: ICML (2019)
- 2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
- 3. Brauwers, G., Frasincar, F.: A general survey on attention mechanisms in deep learning. IEEE Transactions on Knowledge and Data Engineering (2021)
- 4. Chen, J., Ma, T., Xiao, C.: Fastgcn: fast learning with graph convolutional networks via importance sampling. In: ICLR (2018)
- 5. Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional networks. In: ICML (2020)
- 6. Cheng, H.T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., Shah, H.: Wide & deep learning for recommender systems. DLRS (2016)
- 7. Dai, H., Dai, B., Song, L.: Discriminative embeddings of latent variable models for structured data. In: ICML (2016)
- 8. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: SIGKDD (2017)
- 9. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD (2016)
- 10. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS (2017)
- 11. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. In: NeurIPS (2020)
- 12. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous graph transformer. In: WWW (2020)
- 13. Huang, W., Zhang, T., Rong, Y., Huang, J.: Adaptive sampling towards fast graph representation learning. In: NeurIPS (2018)
- 14. Ji, Y., Yin, M., Yang, H., Zhou, J., Zheng, V.W., Shi, C., Fang, Y.: Accelerating large-scale heterogeneous interaction graph embedding learning via importance sampling. TKDD (2020)
- 15. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2016)
- 16. Li, G., Müller, M., Ghanem, B., Koltun, V.: Training graph neural networks with 1000 layers. In: ICML (2021)
- 17. Li, G., Xiong, C., Thabet, A., Ghanem, B.: Deepergen: All you need to train deeper gens. arXiv preprint arXiv:2006.07739 (2020)
- Li, H., Liu, Y., Li, Y., Huang, B., Zhang, P., Zhang, G., Zeng, X., Deng, K., Chen, W., He, C.: Graphtheta: A distributed graph neural network learning system with flexible training strategy (2021), https://arxiv.org/abs/2104.10569
- 19. Liao, L., He, X., Zhang, H., Chua, T.S.: Attributed social network embedding. IEEE Transactions on Knowledge and Data Engineering 30(12), 2257–2270 (2018)
- 20. Liu, M., Gao, H., Ji, S.: Towards deeper graph neural networks. In: SIGKDD (2020)
- Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., Qi, Y.: Geniepath: Graph neural networks with adaptive receptive paths. In: AAAI (2019)
- 22. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)

- Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: EMNLP (2015)
- 24. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. vol. 26, pp. 3111–3119 (2013)
- Niu, Z., Zhong, G., Yu, H.: A review on the attention mechanism of deep learning. Neurocomputing 452, 48–62 (2021)
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, highperformance deep learning library. In: NeurIPS (2019)
- 27. Pavlopoulos, J., Malakasiotis, P., Androutsopoulos, I.: Deeper attention to abusive user content moderation. In: EMNLP (2017)
- 28. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD (2014)
- Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference (2018)
- 30. Shen, T., Jiang, J., Zhou, T., Pan, S., Long, G., Zhang, C.: Disan: Directional self-attention network for rnn/cnn-free language understanding. AAAI (2018)
- 31. Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y.: Masked label prediction: Unified message passing model for semi-supervised classification. arXiv preprint arXiv:2009.03509 (2020)
- 32. Sun, C., Wu, G.: Adaptive graph diffusion networks with hop-wise attention. arXiv preprint arXiv:2012.15024 (2020)
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (2017)
- 34. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
- 35. Wang, G., Ying, R., Huang, J., Leskovec, J.: Direct multi-hop attention based graph neural network. arXiv preprint arXiv:2009.14332 (2020)
- 36. Wang, M., Yu, L., Zheng, D., Gan, Q., Gai, Y., Ye, Z., Li, M., Zhou, J., Huang, Q., Ma, C., et al.: Deep graph library: Towards efficient and scalable deep learning on graphs. In: ICLR (2019)
- 37. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: WWW (2019)
- 38. Wang, Y.: Bag of tricks of semi-supervised classification with graph neural networks. arXiv preprint arXiv:2103.13355 (2021)
- 39. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2018)
- 40. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.i., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: ICML (2018)
- 41. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies. pp. 1480–1489 (2016)
- 42. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: SIGKDD. pp. 974–983 (2018)
- 43. Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J.: Graph transformer networks. In: NeurIPS (2019)

- 44. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: SIGKDD (2019)
- 45. Zhang, C., Swami, A., Chawla, N.V.: Shne: Representation learning for semantic-associated heterogeneous networks. In: WSDM (2019)
- 46. Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., Gu, Q.: Layer-dependent importance sampling for training deep and large graph convolutional networks. In: NeurIPS (2019)