RESEARCH ARTICLE

Explicit Feature Interaction-Aware Graph Neural Network

MINKYU KIM [1], **HYUN-SOO CHOI** [2], **AND JINHO KIM** [3], **(Member, IEEE)**

[1] Department of Research and Development, Ziovision Company Ltd., Chuncheon 24341, Republic of Korea
[2] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea
[3] Department of Computer Science and Engineering, Kangwon National University, Chuncheon 24341, Republic of Korea

Corresponding authors: Hyun-Soo Choi (choi.hyunsoo@seoultech.ac.kr) and Jinho Kim (jhkim@kangwon.ac.kr)

ABSTRACT Graph neural networks (GNNs) are powerful tools for handling graph-structured data. However, their design often limits them to learning only higher-order feature interactions, leaving low-order feature interactions overlooked. To address this problem, we introduce a novel GNN method called explicit feature interaction-aware graph neural network (EFI-GNN). Unlike conventional GNNs, EFI-GNN is a multilayer linear network designed to model arbitrary-order feature interactions explicitly within graphs. To validate the efficacy of EFI-GNN, we conduct experiments using various datasets. The experimental results demonstrate that EFI-GNN has competitive performance with existing GNNs, and when a GNN is jointly trained with EFI-GNN, predictive performance sees an improvement. Furthermore, the predictions made by EFI-GNN are interpretable, owing to its linear construction. The source code of EFI-GNN is available at https://github.com/gim4855744/EFI-GNN.

INDEX TERMS Graph neural networks, feature interactions, interpretable AI.

I. INTRODUCTION

Graphs are ubiquitous in the real world. Molecules, social networks, citation networks, and natural languages are representative examples. Directly analyzing such complex graphs is challenging, and embedding graphs(or nodes) into real vectors is necessary for applying machine learning techniques. As a result, graph-representation learning has emerged as a crucial task in recent years.

Graph neural networks (GNNs) are attracting great attention across various domains due to their remarkable ability to handle graph-structured data [9], [15], [18]. GNNs aggregate input features and the representations of neighboring nodes in nonlinear ways utilizing an activation function. This approach enables the implicit learning of higher-order feature interactions, which can be advantageous in capturing intricate patterns. However, some valuable patterns can be captured from low-order feature interactions (e.g., 1st-, 2nd-, or 3rd-order interactions) [6], [12], [24]. By focusing solely on

higher-order feature interactions, GNNs might miss essential patterns, potentially resulting in a degradation of predictive performance.

Explicitly learning feature interactions- a way humans can understand how features are combined and identify the influence of these interactions on output values- has been widely studied [6], [12], [24], [35]. To learn low-order feature interactions, existing methods employ linear regression [6], factorization [12], [28], or feature crossing [17], [24], [35]. They have also shown that combining low-order and higher-order feature interactions can enhance predictive performance in tasks like recommendations [12], [24] and regression [17]. However, to the best of our knowledge, there are no explicit feature interaction methods that can directly handle graphs. In addition, although explicit methods are inherently interpretable, their interpretability has been neglected in the previous literature.

To overcome the aforementioned problem, we propose a novel GNN method named explicit feature interaction-aware graph neural network (EFI-GNN), which can explicitly learn arbitrary-order feature interactions on a graph.

The associate editor coordinating the review of this manuscript and approving it for publication was Michele Magno.

EFI-GNN is a multilayer linear network. Specifically, EFI-GNN performs graph convolution [18] without an activation function and multiplies the sum of 1st-order features across each layer, which we refer to as feature crossing. Therefore, the interaction order of EFI-GNN escalates with the increase in its number of layers. For instance, 2- and 4-layer EFI-GNNs can learn 2nd- and 4th-order feature interactions, respectively. In addition, EFI-GNN is inherently interpretable due to its linearity.

We conduct experiments using various citation network datasets to validate the effectiveness of EFI-GNN. The experimental results demonstrate that EFI-GNN has competitive predictive performance with existing GNNs. Moreover, jointly training an existing GNN with EFI-GNN can improve predictive performance. To determine the efficacy of learning multiple feature interactions in graphs, we conduct further experiments across various joint-learning scenarios using open graph benchmark (OGB) datasets. The experimental results demonstrate that combining multiple feature interactions leads to performance improvements. This implies that sophisticated techniques, previously proposed for learning feature interactions to improve recommendation or regression performances, may be potent for graph applications. In addition, to interpret the prediction of EFI-GNN, we visualize 1st- and 2nd-order feature interactions as heatmaps.

II. RELATED WORKS
A. GRAPH NEURAL NETWORKS
GNNs are a special form of deep neural networks tailored for graph-structured data. Vanilla graph neural network (VanillaGNN) [30] extends recurrent neural networks (RNNs) [29] to make them applicable to more general graphs, like undirected cyclic graphs. Gated GNN (GGNN) [21] improves VanillaGNN by using gated recurrent unit (GRU) [7]. These RNN-based GNNs can be classified into Recurrent GNNs (RGNNs). Planetoid [39] is a semi-supervised node embedding framework trained to predict node class labels and subgraph contexts simultaneously through a neural network. Recently, graph convolutional network (GCN) [18] and its variants [13], [34] have achieved great success in the node embedding task. One problem with spectral methods like GCN is over-smoothing. To alleviate the over-smoothing problem and train a deep GCN, several methods utilizing residual connections have been proposed [19], [20]. Jumping knowledge network (JKNet) [38] concatenates all hidden layers of GCN to make predictions. This strategy can alleviate the over-smoothing problem. Subgraph sampling techniques for fast training and inference of GCN have been proposed [3], [15]. Simple graph convolutional network (SGC) [36] is a straightforward and linear version of GCN using powers of the adjacency matrix. Link prediction methods for bipartite graphs utilizing GNNs have been proposed [1], [41]. Graph transformer [40] automatically generates meta-paths and effectively deals with heterogeneous graphs. All the aforementioned methods except SGC implicitly aggregate feature information. Thus, they only learn higher-order feature interactions and cannot capture valuable patterns occurring in low-order feature interactions.

B. FEATURE INTERACTION METHODS
Feature interactions refer to the combined effects of different features on a given output value. Such interactions play a crucial role in improving the performances of machine learning models [12], [28]. Traditionally, handcrafted features have been widely used to learn feature interactions. However, generating handcrafted features requires domain expertise and is time-consuming. Recently, various methods have been proposed for learning feature interactions without handcrafting. One example is factorization machine (FM) [28], which combines linear regression and factorization methods to learn 1st- and 2nd-order feature interactions simultaneously. Similarly, Wide & deep learning [6] combines linear regression and deep neural network (DNN) to learn 1st- and higher-order feature interactions simultaneously. DeepFM [12] is a joint learning method of FM and DNN. The aforementioned models learn only bounded-order feature interactions. To overcome this problem, CrossNet [35] has been proposed. It can learn arbitrary-order feature interactions based on its number of layers. Compressed interaction network (CIN) [24] and the explicit component of extreme interaction network (XIN) [17] have extended CrossNet to vector-wise operations.

C. FEATURE INTERACTIONS VIA GRAPHS
In recent years, several methods have been proposed for learning feature interactions through GNNs [22], [23], [25], [42]. These methods designate features as nodes and employ GNNs to learn edge weights within the feature graph. The learned edge weights indicate the interactions between different features. However, they cannot handle graph-structured data. FI-GNN [8] is a joint learning method of GNN and the feature factorization module. However, the feature factorization module cannot apply directly to graph-structured data and only learns 2nd-order feature interactions. In contrast, our proposed EFI-GNN can directly handle graph-structured data and learn arbitrary-order feature interactions.

III. EFI-GNN
A. PRELIMINARIES
1) FEATURE INTERACTION
Feature interactions are the influences of features to model outcomes. 1st-order feature interactions indicate the influences of single features. Similarly, 2nd- and 3rd-order feature interactions indicate the influences of 2nd- and 3rd-order features (e.g., user $\times$ movie, user $\times$ movie $\times$ date). More formally, feature interaction can be defined as follows:
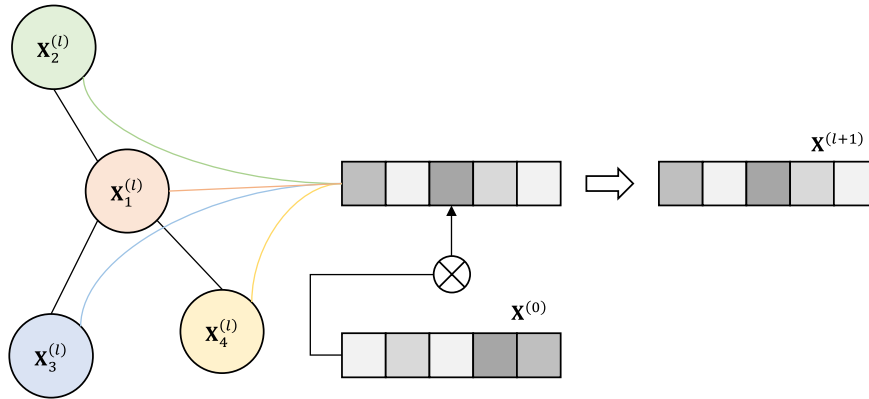
FIGURE 1. EFI-GNN layer.

Definition 1: If a function $h(\cdot)$ holds the following condition, there are interactions between features $\mathbf{x}$ [33].

$$h(x_1 + x_2 + \ldots + x_m) \neq \sum_{i=1}^{m} h(x_i). \tag{1}$$

All deep learning models always hold the above condition due to their non-linearity. Therefore, deep learning models always and only learn interactions between all features (i.e., higher-order feature interactions).

B. EFI-GNN LAYER

EFI-GNN has a multilayer architecture but is quite different from existing GNNs. Each layer in EFI-GNN has a feature-crossing term, which multiplies the sum of 1st-order features instead of an activation function. In EFI-GNN, the sum of 1st-order features is defined as follows:

$$\mathbf{X}^{(1)} = \mathbf{X}^{(0)}\mathbf{W}^{(1)}, \tag{2}$$

where $\mathbf{X}^{(0)} \in \mathbb{R}^{N \times M}$ is the raw feature matrix, $\mathbf{W}^{(1)} \in \mathbb{R}^{M \times K}$ is the trainable weight matrix, N is the number of nodes in the graph, M is the number of features, and K is the projection size. An EFI-GNN layer is defined as follows:

$$\mathbf{X}^{(l)} = \hat{\mathbf{A}}\mathbf{X}^{(l-1)}\mathbf{W}^{(l)} \odot \mathbf{X}^{(1)}, \tag{3}$$

where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix, $\tilde{\mathbf{D}} \in \mathbb{R}^{N \times N}$ is the degree matrix of $\tilde{\mathbf{A}}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{K \times K}$ is the trainable weight matrix in l^{th} layer, and $\odot$ indicates the Hadamard product. The adjacency matrix $\hat{\mathbf{A}}$ is multiplied by the feature matrix $\mathbf{X}^{(l-1)}$ to aggregate the neighbor nodes' information, and the sum of 1st-order features is multiplied in each layer to learn high-order feature interactions. Therefore, the order of feature interactions that EFI-GNN learns gradually escalates with the number of layers increases. Fig. 1 depicts the architecture of an EFI-GNN layer.

Since each layer of EFI-GNN captures different feature interactions, the output layer of EFI-GNN leverages these rich feature interactions to enhance its predictive performance.
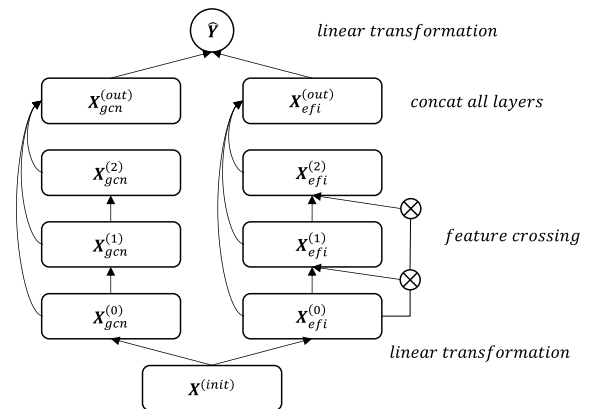

FIGURE 2. The architecture of GCN & EFI-GNN.

The final output layer of EFI-GNN is defined as follows:

$$\hat{\mathbf{Y}} = \left[\mathbf{X}^{(1)} \| \mathbf{X}^{(2)} \| \ldots \| \mathbf{X}^{(L)}\right]\mathbf{W}^{(\text{out})}, \tag{4}$$

where $\|$ is the concatenation operator, L is the number of layers, $\mathbf{W}^{(\text{out})} \in \mathbb{R}^{K \cdot L \times O}$ is the trainable output matrix, and O is the output size.

C. EFI-GNN LAYER ANALYSIS

To reveal how the EFI-GNN layer captures feature interactions, we present the vector-level representation of the layer. In (2), $\mathbf{X}^{(1)}$ represents the linear transformation of the input features. In other words, $\mathbf{X}^{(1)}$ indicates the weighted sum of 1st-order features. Therefore, we can express (2) at the vector level as:

$$\mathbf{x}_v^{(1)} = \left(x_{v,1}^{(0)}\mathbf{w}_1^{(1)} + x_{v,2}^{(0)}\mathbf{w}_2^{(1)} + \ldots + x_{v,M}^{(0)}\mathbf{w}_M^{(1)}\right)$$
$$= \sum_{m=1}^{M} x_{v,m}^{(0)}\mathbf{w}_m^{(1)}, \tag{5}$$

where v denotes the node index. (5) shows that $\mathbf{x}_v^{(1)}$ is the weighted sum of the input features of node v. In (3), $\mathbf{X}^{(l)}$ stands for the linear transformation of $\mathbf{X}^{(l-1)}$,

TABLE 1. Dataset statistics.

	Cora	CiteSeer	PubMed	Amazon	Actor	ogbn-arxiv	ogbn-mag
# of Nodes	2,708	3,327	19,717	13,752	7,600	169,343	1,939,743
# of Edges	5,429	4,732	44,338	491,722	30,019	1,166,243	21,111,007
# of Features	1,433	3,703	500	767	932	128	128
# of Classes	7	6	3	10	5	40	349
# of Training Nodes	1,208	1,827	18,217	8,252	3,648	90,941	629,571
# of Validation Nodes	500	500	500	2,750	2,432	29,799	64,879
# of Test Nodes	1,000	1,000	1,000	2,750	1,520	48,603	41,939
Graph Type	undirected	undirected	undirected	undirected	undirected	directed	directed
Feature Type	zero/one	zero/one	tf/idf	zero/one	zero/one	word2vec	word2vec

TABLE 2. Hyperparameter settings for the experimental datasets.

	Cora	CiteSeer	PubMed	Amazon	Actor	OGB
# of GNN Layers	3	3	3	3	3	1
# of EFI-GNN Layers	2	2	2	2	2	1
# of Units	128	128	1024	128	128	128
Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3	1e-2
L2 Penalty	1e-2	1e-2	1e-3	1e-2	1e-2	No
Dropout	0.9	0.9	0.85	0.9	0.9	0.3
Activation	LeakyReLU	LeakyReLU	LeakyReLU	LeakyReLU	LeakyReLU	LeakyReLU
Residual Connection	No	No	Dense	No	No	No
Batch Normalization	No	No	Yes	No	No	Yes

achieved through neighbor aggregation and feature-crossing. Therefore, (3) can be expressed at the vector level as:

$$\mathbf{x}_v^{(l)} = \left(\hat{\mathbf{A}} \mathbf{X}^{(l-1)} \mathbf{W}^{(l)} \right)_v \odot \mathbf{x}_v^{(1)}$$

$$= \sum_{i \in \mathcal{N}(v)} \left(x_{i,1}^{(l-1)} \mathbf{w}_1^{(l)} + x_{i,2}^{(l-1)} \mathbf{w}_2^{(l)} + \ldots + x_{i,K}^{(l-1)} \mathbf{w}_K^{(l)} \right)$$

$$\odot \left(x_{v,1}^{(0)} \mathbf{w}_1^{(1)} + x_{v,2}^{(0)} \mathbf{w}_2^{(1)} + \ldots + x_{v,M}^{(0)} \mathbf{w}_M^{(1)} \right)$$

$$= \sum_{i \in \mathcal{N}(v)} \sum_{k=1}^{K} x_{i,k}^{(l-1)} \mathbf{w}_k^{(l)} \odot \sum_{m=1}^{M} x_{v,m}^{(0)} \mathbf{w}_m^{(1)}. \quad (6)$$

In (6), the previous layer, $\sum_{i \in \mathcal{N}(v)} \sum_{k=1}^{K} x_{i,k}^{(l-1)} \mathbf{w}_k^{(l)}$, is multiplied by the sum of 1st-order features, $\sum_{m=1}^{M} x_{v,m}^{(0)} \mathbf{w}_m^{(1)}$. This allows EFI-GNN layers to learn high-order feature interactions gradually.

As highlighted above, feature crossing is a pivotal component of our EFI-GNN. Standard linear models possess limited expressive power regardless of their number of layers. This limitation arises because they can only learn 1st-order feature interactions and linear patterns. However, linear models with feature crossing can learn high-order feature interactions, and this strategy allows linear models to partially overcome the limitation [24], [35].

D. INTERPRETABILITY

If an EFI-GNN layer is directly connected to the output layer, we can obtain the influences of features within that layer, leveraging the intrinsically linear nature of EFI-GNN. We describe the procedure for obtaining feature influences below. The influence of a 1st-order feature is defined as follows:

$$\mathbf{a}_{n,i}^{(1)} = x_{n,i}^{(0)} \mathbf{w}_i^{(1)}, \quad (7)$$

$$e_{c,n,i}^{(1)} = \mathbf{a}_{n,i}^{(1)} \mathbf{w}_{:k,c}^{(out)}, \quad (8)$$

where $\mathbf{a}_{n,i}^{(1)} \in \mathbb{R}^k$ is the representation vector for the 1st-order feature i of the node n, $\mathbf{w}_i^{(1)} \in \mathbb{R}^k$ denotes the weight vector for feature i, $\mathbf{w}_{:u,c}^{(out)} \in \mathbb{R}^k$ is the output weight vector corresponding to the class c, k denotes the number of units in the layer, and $e_{c,n,i}^{(1)}$ indicates the influence value of the 1st-order feature i in the node n for the class c. Analogously, the influence of a 2nd-order feature is defined as follows:

$$\mathbf{a}_{n,i,j}^{(2)} = \left(x_{n,i}^{(0)} \mathbf{w}_i^{(1)} \odot x_{n,j}^{(0)} \mathbf{w}_j^{(1)} \right) \mathbf{W}^{(2)}$$

$$= \left(\mathbf{a}_{n,i}^{(1)} \odot \mathbf{a}_{n,j}^{(1)} \right) \mathbf{W}^{(2)}, \quad (9)$$

$$e_{c,n,i,j}^{(2)} = \mathbf{a}_{n,i,j}^{(2)} \mathbf{w}_{u:2 \cdot u,c}^{(out)}, \quad (10)$$

where $\mathbf{a}_{n,i,j}^{(2)}$ is the representation vector of the 2nd-order feature $i \times j$, and $e_{c,n,i,j}$ is the influence value of the 2nd-order feature $i \times j$ for the class c. Therefore, we can generalize the above equations as follows:

$$\mathbf{a}_{n,i,\ldots,k}^{(l)} = \left(\mathbf{a}_{n,i,\ldots,j}^{(l-1)} \odot \mathbf{a}_{n,k}^{(1)} \right) \mathbf{W}^{(l)}, \quad (11)$$

$$e_{c,n,i,\ldots,k}^{(l)} = \mathbf{a}_{n,i,\ldots,k}^{(l)} \mathbf{w}_{l \cdot u:(l+1) \cdot u,c}^{(out)}, \quad (12)$$

where $e_{c,n,i,\ldots,k}^{(l)}$ indicates the influence value of a lth-order feature.

E. JOINT LEARNING WITH GNNS

Our EFI-GNN can be jointly trained with another GNN. In this section, we introduce a joint learning approach

TABLE 3. Classification accuracies on the Cora, CiteSeer, PubMed, Amazon, and Actor datasets.

		Cora	CiteSeer	PubMed	Amazon	Actor
reported	GCN [18]	86.64%	79.34%	90.22%	N/A	N/A
	FastGCN [3]	85.00%	77.60%	88.00%	N/A	N/A
	GraphSage [13]	82.20%	71.40%	87.10%	N/A	N/A
	ASGCN [15]	87.44±0.34%	79.66±0.18%	90.60±0.16%	N/A	N/A
	GCNII [4]	88.49%	77.08%	89.57%	N/A	N/A
	EFI-GNN	87.95±0.53%	78.45±0.57%	89.85±0.22%	88.43±0.12%	30.14±0.19%
our	GCN [18]	87.88±0.20%	79.23±0.31%	90.07±0.40%	88.75±0.10%	29.72±0.26%
	+ EFI-GNN	88.60±0.21%	79.80±0.20%	90.87±0.23%	88.98±0.13%	30.28±0.09%
	GAT [34]	86.78±0.22%	78.92±0.31%	88.64±0.32%	83.27±1.81%	29.74±0.21%
	+ EFI-GNN	87.50±0.24%	80.54±0.18%	89.82±0.15%	88.12±0.79%	30.28±0.16%
	GCNII [4]	88.41±0.32%	79.42±0.23%	90.34±0.21%	88.59±0.10%	29.97±0.27%
	+ EFI-GNN	88.77±0.28%	79.54±0.37%	90.41±0.34%	88.76±0.17%	30.00±0.23%
	GATv2 [2]	86.14±0.18%	78.84±0.30%	89.12±0.22%	85.50±2.45%	30.20±0.19%
	+ EFI-GNN	86.88±0.60%	80.50±0.19%	90.02±0.32%	87.61±0.63%	30.12±0.36%
	ASDGN [11]	85.68±0.29%	78.42±0.19%	88.68±0.36%	82.81±0.96%	29.68±0.28%
	+ EFI-GNN	87.86±0.45%	79.60±0.07%	89.58±0.24%	88.47±0.12%	30.20±0.32%

TABLE 4. Classification accuracies on the OGB datasets.

	ogbn-arxiv	ogbn-mag
GCN*	56.61±0.15%	26.35±0.25%
EFI-GNN*	56.82±0.26%	25.82±0.29%
GCN	56.94±0.07%	26.58±0.17%
GAT	57.20±0.20%	25.75±0.20%
GATv2	56.26±0.19%	25.76±0.19%
ASDGN	56.72±0.30%	25.78±0.11%
EFI-GNN	57.37±0.28%	26.25±0.24%
GCN & GCN	57.40±0.15%	26.58±0.38%
EFI-GNN & EFI-GNN	57.45±0.19%	26.25±0.19%
GCN & EFI-GNN	57.96±0.13%	26.79±0.22%
GAT & EFI-GNN	57.25±0.26%	25.87±0.18%
GATv2 & EFI-GNN	56.56±0.11%	26.00±0.27%
ASDGN & EFI-GNN	57.45±0.26%	26.38±0.10%

that combines EFI-GNN and an existing GNN. Each layer of EFI-GNN explicitly learns different-order feature interactions, while each layer of an existing GNN implicitly learns higher-order feature interactions - with upper layers capturing more complex patterns than lower layers. Since both EFI-GNN and GNN capture different patterns at each layer, utilizing these rich feature interactions can improve predictive performances. The final representation matrices of both EFI-GNN and GNN are defined as follows:

$$\mathbf{X}_{efi}^{(out)} = \left[\mathbf{X}_{efi}^{(1)} \parallel \mathbf{X}_{efi}^{(2)} \parallel \ldots \parallel \mathbf{X}_{efi}^{(L_{efi})} \right], \quad (13)$$

$$\mathbf{X}_{gnn}^{(out)} = \left[\mathbf{X}_{gnn}^{(1)} \parallel \mathbf{X}_{gnn}^{(2)} \parallel \ldots \parallel \mathbf{X}_{gnn}^{(L_{gnn})} \right], \quad (14)$$

where $\mathbf{X}_{efi}^{(out)} \in \mathbb{R}^{N \times K \cdot L_{efi}}$ and $\mathbf{X}_{gnn}^{(out)} \in \mathbb{R}^{N \times K \cdot L_{gnn}}$ are the final representation vectors of EFI-GNN and GNN, respectively, and L_{efi} and L_{gnn} are the numbers of layers of EFI-GNN and GNN, respectively. The final output layer of the joint-learning method is defined as follows:

$$\hat{\mathbf{Y}} = \left[\mathbf{X}_{efi}^{(out)} \parallel \mathbf{X}_{gnn}^{(out)} \right] \mathbf{W}^{(out)}, \quad (15)$$

where $\mathbf{W}^{(out)} \in \mathbb{R}^{K \cdot (L_{efi} + L_{gnn}) \times O}$ is the trainable output matrix. We use the cross-entropy as the loss function defined as:

$$\mathcal{L} = - \sum_{i \in \mathcal{V}} \mathbf{y}_i log \hat{\mathbf{y}}_i, \quad (16)$$

where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of nodes.

IV. EXPERIMENTS

A. DATASETS

We compare the performance of the proposed EFI-GNN and the joint-learning methods against other leading GNNs using three citation network datasets: Cora, CiteSeer, and PubMed, as well as the Amazon Computers [31] and Actor [27] datasets. In the Amazon, nodes indicate products in the Computers section and edges represent that two products are frequently bought together. In the Actor, each node indicates an actor, and an edge denotes the co-occurrence of two actors on the same Wikipedia page. In addition, we conduct experiments with two open graph benchmark (OGB) datasets to explore whether combining multiple feature interactions effectively improves predictive performance. The statistics of the experimental datasets are outlined in Table 1. The objectives of all datasets are to predict the classes of nodes. The obgn-mag is a heterogeneous network. Thus, we convert it to a homogeneous network by focusing solely on the (paper, cites, paper) relationships. For the Cora, CiteSeer, Amazon, and Actor, nodes' features are zero/one encoded vectors representing absent/present words. For the PubMed, nodes' features represent TF/IDF score vectors of the words. For the obgn-arxiv and ogbn-mag, nodes' features represent the average of word2vec vectors of the present words. All experimental datasets are publicly available. For the Cora, CiteSeer, and PubMed, we split the datasets into training/validation/test sets with the same scenario as [3]. For the Amazon and Actor, we randomly split the datasets
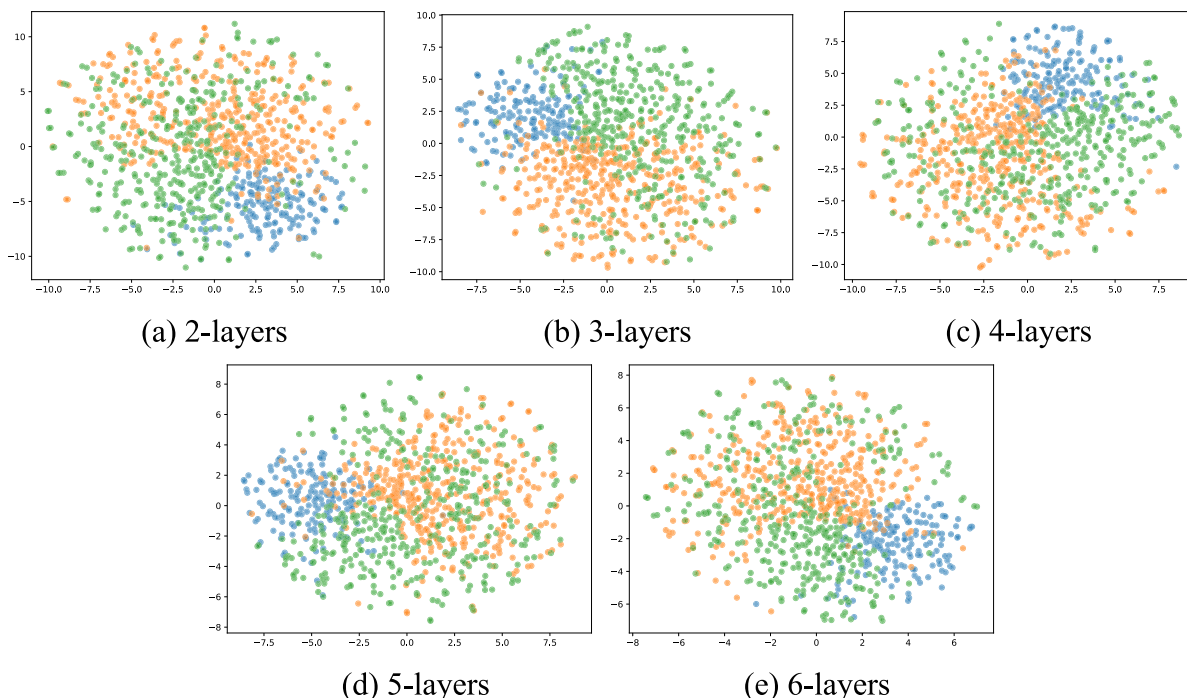
(a) 2-layers

(b) 3-layers

(c) 4-layers

(d) 5-layers

(e) 6-layers

FIGURE 3. tSNE results of EFI-GNNs' final representations on the PubMed.
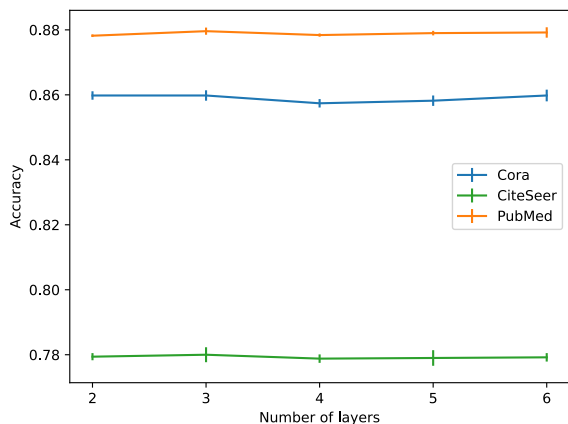

FIGURE 4. Ablation study on the number of layers for the Cora, CiteSeer, and PubMed.

in training/validation/test sets with 60%/20%/20% ratio. For the obgn-arxiv and ogbn-mag, we split the datasets into training/validation/test sets with the same scenario as the public OGB test.

B. EXPERIMENTAL SETUP

We implemented the proposed EFI-GNN and joint-learning methods using PyTorch [26] and PyTorch Geometric [10]. We trained the EFI-GNN, GCN, and GCNII models on an NVIDIA GTX 1080ti with 12GB VRAM, while the remaining models, namely GAT, GATv2, and ASDGN, were trained on an NVIDIA A100 with 80GB VRAM. Training the

GAT, GATv2, and ASDGN models on the GTX 1080ti for the ogbn-mag dataset is infeasible due to their high memory usage. Specifically, our PyTorch Geometric implementations of these models require 30GB, 42GB, and 18GB of VRAM, respectively.

For all experiments, L2 penalty and dropout are applied [32]. For the PubMed, ogbn-arxiv, and ogbn-mag, batch normalization [16] is used. To expedite the approximation speed of the models on PubMed, the residual connection [14] is employed. We optimize hyperparameters, including learning rate and number of units for the Cora, CiteSeer, PubMed, Amazon, and Actor, using grid search. For a fair comparison, we use the same model structure and hyperparameters for ogbn-arxiv and ogbn-mag. All experimental models were trained 200 epochs for Cora, CiteSeer, and PubMed, while they were trained 1000 epochs for the ogbn-arxiv and ogbn-mag. Cross entropy is used as the loss function, and trainable parameters are optimized with Adam optimizer. Table 2 shows the hyperparameter settings for experimental datasets.

C. PERFORMANCE COMPARISON

We compare EFI-GNN with GCN [18], FastGCN [3], GraphSage [13], ASGCN [15], GCNII [4], GAT [34], GATv2 [2], and ASDGN [11] using the Cora, CiteSeer, PubMed, Amazon, and Actor. Table 3 shows the test set accuracies for experimental models. In the *"reported"* row of Table 3, we have borrowed the reported performances of GCN, FastGCN, GraphSage, ASGCN, and GCNII from their original papers. In addition, we conducted our own

experiments with GCN, GAT, GCNII, GATv2, and ASDGN. We measured the accuracies of the experimental models with 10 random seeds and reported the average accuracies and standard deviations. Our experimental models concatenate all layers to make predictions. Although our EFI-GNN is linear, it shows comparable performance to GCN. This demonstrates that EFI-GNN can effectively learn feature interactions on graphs. In addition, the predictive accuracies are consistently improved in the all joint-learning settings. This shows that the low-order feature interactions learned by EFI-GNN include useful information that cannot be captured in higher-order interactions.

We conduct an additional experiment on two OGB datasets to examine whether combining multiple feature interactions is effective. In this experiment, GCN, GAT, GATv2, and ASDGN are used to learn implicit feature interactions, and EFI-GNN is used to learn explicit feature interactions. Table 4 shows the test accuracies of the experimental models. GCN* and EFI-GNN* indicate GCN and EFI-GNN that make predictions using only the last layer, respectively. GCN, GAT, GATv2, and ASDGN use the concatenation of all layers to make predictions. Thus, they can learn multiple implicit higher-order feature interactions. EFI-GNN also uses the concatenation of all layers to make predictions. It can learn multiple various-order feature interactions. All joint learning methods can simultaneously learn multiple implicit higher-order and explicit various-order feature interactions. For a fair comparison, we use GCN & GCN, which is a joint-learning method of two GCNs, and EFI-GNN & EFI-GNN, which is the joint-learning method of two EFI-GNNs. They have the same number of trainable parameters as GCN & EFI-GNN. GCN and EFI-GNN outperform GCN* and EFI-GNN*, respectively. This shows that simultaneously learning multiple feature interactions is effective for predictive performance. Furthermore, GCN & EFI-GNN outperforms GCN & GCN and EFI-GNN & EFI-GNN, and the performance of other joint learning methods also consistently improved. This demonstrates that simultaneously learning both explicit and implicit feature interactions is more effective than learning only explicit or implicit ones.

D. ABLATION STUDY

Deep spectral graph neural networks may cause the over-smoothing problem. To investigate whether EFI-GNN faces the same problem, we have assessed the performance of EFI-GNN on the Cora, CiteSeer, and PubMed while varying the number of layers. Fig. 4 shows the performance changes as the number of layers changes. In the experiment, we have observed no significant changes in performance. In Fig. 3, we have visualized the tSNE results of the EFI-GNNs' final representations on the PubMed dataset. Fig. 3 shows that EFI-GNN does not suffer from over-smoothing problems. This stability can be caused by the jumping knowledge mechanism and residual connections of EFI-GNN. Some previous works have demonstrated that

the jumping knowledge mechanism and residual connections help avoid the over-smoothing problem [4], [19], [20], [38].
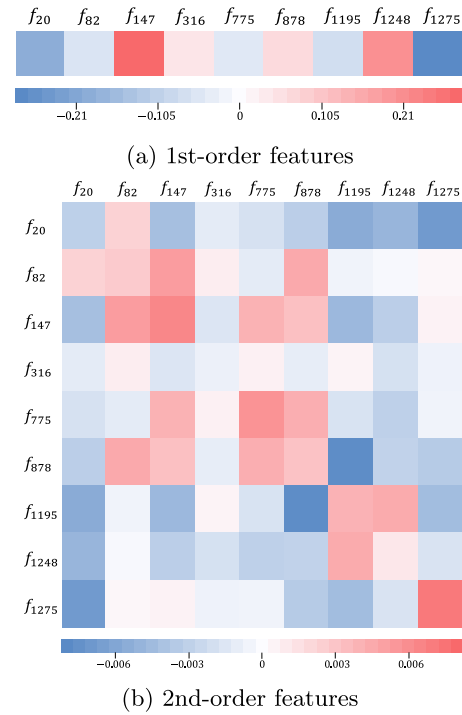

(a) 1st-order features

(b) 2nd-order features

FIGURE 5. Influences of 1st- and 2nd-order features.

(a) Units 32

(b) Units 64

(c) Units 128

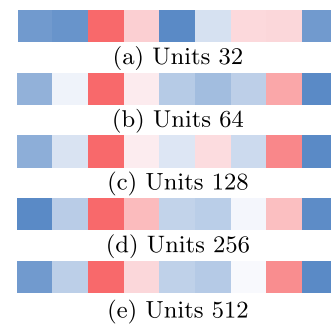(d) Units 256

(e) Units 512

FIGURE 6. First-order feature interactions with the different number of units.

E. VISUALIZATION OF FEATURE INTERACTIONS

Our EFI-GNN is intrinsically interpretable, allowing us to discern the influences of features from a trained model. Fig. 5 illustrates the influences of both 1st- and 2nd-order features on node 1 with respect to class 1 in the Cora dataset. Since the node has nine active features, we visualized the influences of these particular features. In the figure, red cells represent features with positive influences, while blue cells represent features with negative influences. Positive features increase the probability of the node label being 1, whereas negative features decrease the probability.

We demonstrated the interpretability of our EFI-GNN by visualizing its feature interactions. In this section,
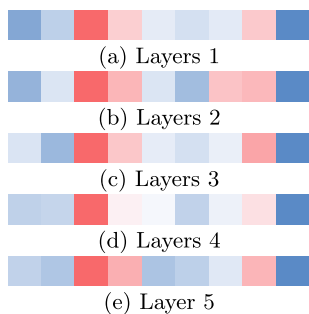
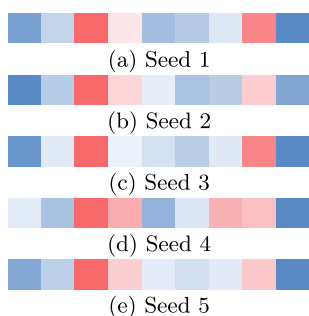**FIGURE 7.** First-order feature interactions with the different number of layers.

FIGURE 8. First-order feature interactions with the different random seeds.
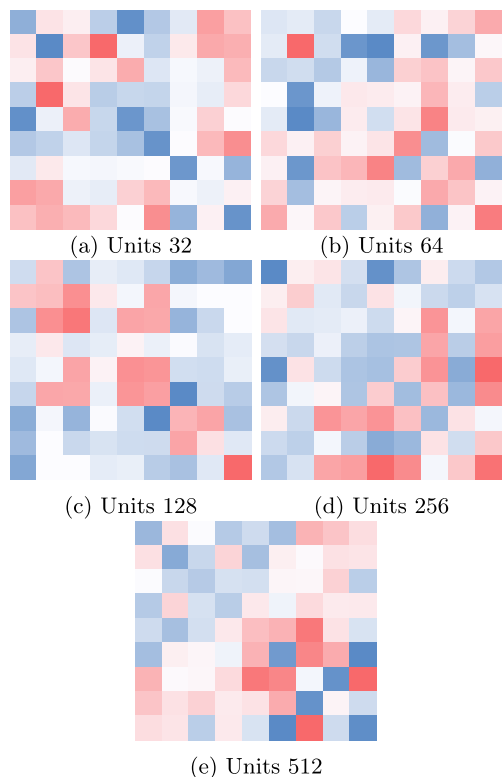

FIGURE 9. Second-order feature interactions with the different number of units.

we additionally provide visualizations of 1st- and 2nd-order feature interactions across various hyper-parameter settings. Fig. 6, Fig. 7, and Fig. 8 show the 1st-order feature

interactions for different numbers of units, layers, and random seeds, respectively. As shown in the figures, the 1st-order interactions have a consistent trend across different hyper-parameter settings. It indicates that EFI-GNN pays attention to the same features and shows its reliability. On the other hand, Fig. 9 shows the 2nd-order feature interactions across different numbers of units. Unlike the consistent trends observed in the 1st-order interactions, it is hard to find any patterns in the 2nd-order interactions. We conjecture that the node aggregation in EFI-GNN layer causes this problem. This aggregation may make it more challenging to capture stable patterns of feature interactions. Our future work is to find out why this phenomenon is caused.

V. CONCLUSION

In this paper, we proposed EFI-GNN, a novel GNN architecture. EFI-GNN is a linear and interpretable model that can explicitly learn arbitrary-order feature interactions on graphs. EFI-GNN showed comparable performance to GCN, and combining EFI-GNN with another GNN further enhanced predictive performance. In addition, we visualized the influences of the 1st- and 2nd-order features through heatmaps. We believe that EFI-GNN is a pioneering work for learning feature interactions on graph-structured data.

The major limitations of EFI-GNN are as follows: 1) Interpreting feature interactions more than 2nd-order is challenging due to neighbor aggregation. 2) EFI-GNN can only deal with homogeneous graphs. 3) Analysis of EFI-GNN behavior on homophilic and heterophilic graphs should be conducted. Based on these limitations, our avenues for future works include: 1) extending EFI-GNN to other domains where feature interactions play an important role, such as recommender systems; 2) incorporating sophisticated feature interaction learning techniques into EFI-GNN; 3) deploying EFI-GNN in high-stakes domains that necessitate interpretable models; and 4) applying EFI-GNN to other types of graphs such as heterogeneous or dynamic graphs.

REFERENCES

[1] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2017, *arXiv:1706.02263*.

[2] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022.

[3] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[4] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1725–1735.

[5] Z. Chen, L. Fu, J. Yao, W. Guo, C. Plant, and S. Wang, "Learnable graph convolutional network and feature fusion for multi-view learning," *Inf. Fusion*, vol. 95, pp. 109–119, Jul. 2023.

[6] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.

[7] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.

[8] K. Ding, Y. Li, J. Li, C. Liu, and H. Liu, "Feature interaction-aware graph neural networks," 2019, *arXiv:1908.07110*.

[9] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *Proc. World Wide Web Conf.*, May 2019, pp. 417–426.

[10] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," in *Proc. ICLR Workshop, Represent. Learn. Graphs Manifolds*, 2019.

[11] A. Gravina, D. Bacciu, and C. Gallicchio, "Anti-symmetric DGN: A stable architecture for deep graph networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2023.

[12] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization machine based neural network for CTR prediction," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1725–1731.

[13] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[15] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4563–4572.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[17] M. Kim, S. Lee, and J. Kim, "Combining multiple implicit-explicit interactions for regression analysis," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 74–83.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[19] G. Li, M. Müller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9267–9276.

[20] G. Li, C. Xiong, A. Thabet, and B. Ghanem, "DeeperGCN: All you need to train deeper GCNs," 2020, *arXiv:2006.07739*.

[21] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.

[22] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, "Fi-GNN: Modeling feature interactions via graph neural networks for CTR prediction," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 539–548.

[23] Z. Li, S. Wu, Z. Cui, and X. Zhang, "GraphFM: Graph factorization machines for feature interaction modeling," 2021, *arXiv:2105.11866*.

[24] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "XDeepFM: Combining explicit and implicit feature interactions for recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1754–1763.

[25] Y. Liu, C. Li, H. Xiao, and J. Cai, "GCN-Int: A click-through rate prediction model based on graph convolutional network interaction," *IEEE Access*, vol. 9, pp. 140022–140030, 2021.

[26] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[27] H. Pei, B. Wei, K. Chen-Chuan Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric graph convolutional networks," 2020, *arXiv:2002.05287*.

[28] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 995–1000.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Dec. 2009.

[31] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2018, *arXiv:1811.05868*.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, Jun. 2014.

[33] M. Tsang, S. Rambhatla, and Y. Liu, "How does this interaction affect me? Interpretable attribution for feature interactions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6147–6159.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.

[35] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proc. ADKDD*, Aug. 2017, pp. 1–7.

[36] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.

[37] Z. Wu, X. Lin, Z. Lin, Z. Chen, Y. Bai, and S. Wang, "Interpretable graph convolutional network for multi-view semi-supervised learning," *IEEE Trans. Multimedia*, vol. 25, pp. 8593–8606, Mar. 2023.

[38] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 5453–5462.

[39] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 40–48.

[40] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 11983–11993.

[41] M. Zhang and Y. Chen, "Inductive matrix completion based on graph neural network," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020.

[42] Y. Zheng, P. Wei, Z. Chen, Y. Cao, and L. Lin, "Graph-convolved factorization machines for personalized recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 2, pp. 1567–1580, Feb. 2023.

MINKYU KIM received the B.S. degree in computer science and the M.S. degree in big data medical convergence from Kangwon National University, Chuncheon, Republic of Korea, in 2020 and 2022, respectively.

Since 2022, he has been a Researcher with Ziovision Inc., Chuncheon. His research interests include clinical decision support systems, recommender systems, graph neural networks, and explainable/interpretable AI.

HYUN-SOO CHOI received the B.S. degree in computer and communication engineering for the first major and in brain and cognitive science for the second major from Korea University, in 2013, and the integrated M.S./Ph.D. degree in electrical and computer engineering from Seoul National University, South Korea, in 2020. From 2020 to 2021, he was a Senior Researcher with Vision AI Labs, SK Telecom. He was an Assistant Professor with Kangwon National University, South Korea, in 2021. Since October 2021, he has been a Chief Technical Officer with Ziovision. Since 2023, he has also been an Assistant Professor with the Seoul National University of Science and Technology.

JINHO KIM (Member, IEEE) received the B.S. degree in computer engineering from Kyungpook National University, South Korea, in 1982, and the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology, South Korea, in 1985 and 1990, respectively.

He was a Visiting Scholar with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA, from 1995 to 1996, the College of Computing and Informatics, Drexel University, Philadelphia, PA, USA, from 2003 to 2004, and the School of Computing, Georgia Institute of Technology, Atlanta, GA, USA, from 2009 to 2010, respectively. He is currently a Professor with the Department of Computer Science, Kangwon National University, Chuncheon, South Korea. His research interests include high performance database systems, Hadoop/MapReduce parallel and distributed processing technology, cloud computing, data warehousing and OLAP, big data analytics, graph neural networks, and medical/healthcare data analytics.

● ● ●