IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 30, 2021, accepted October 16, 2021, date of publication October 21, 2021, date of current version October 28, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3121708

Graphs, Entities, and Step Mixture for Enriching Graph Representation

KYUYONG SHIN[1], WONYOUNG SHIN[2], JUNG-WOO HA[1], (Member, IEEE), AND SUNYOUNG KWON[3], (Member, IEEE)

[1]NAVER AI Laboratory, NAVER CLOVA, Seongnam 13561, South Korea
[2]NAVER Shopping, Naver Corporation, Seongnam 13561, South Korea
[3]School of Biomedical Convergence Engineering, Pusan National University, Yangsan 50612, South Korea

Corresponding author: Sunyoung Kwon (skwon@pusan.ac.kr)

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) Grant funded by the Korea Government (MSIT) under Grant 2020-0-01450 (Artificial Intelligence Convergence Research Center [Pusan National University]); in part by the BK21 FOUR, Korean Southeast Center for the 4th Industrial Revolution Leader Education; and in part by the Pusan National University Research Grant, 2020.

ABSTRACT Graph neural networks have shown promising results on representing and analyzing diverse graph-structured data, such as social networks, traffic flow, drug discovery, and recommendation systems. Existing approaches for graph neural networks typically suffer from the oversmoothing issue that results in indistinguishable node representation, as recursive and simultaneous neighborhood aggregation deepens. Also, most methods focus on transductive scenarios that are limited to fixed graphs, which do not generalize properly to unseen graphs. To address these issues, we propose a novel graph neural network that considers both edge-based neighborhood relationships and node-based entity features with multiple steps, i.e. **G**raph **E**ntities with **S**tep **M**ixture via *random walk* (GESM). GESM employs a mixture of various steps through random walk to alleviate the oversmoothing problem, an attention mechanism to dynamically reflect interrelations depending on node information, and structure-based regularization to enhance embedding representation. With intensive experiments, we show that our proposed GESM achieves state-of-the-art or comparable performances on eight benchmark graph datasets in both transductive and inductive learning tasks. Furthermore, we empirically demonstrate the superiority of our method on the oversmoothing issue with rich graph representations. Our source code is available at https://github.com/ShinKyuY/GESM.

INDEX TERMS Graph neural network, random walk, oversmoothing, aggregation, mixture, attention, graph, embedding, representation, regularization.

I. INTRODUCTION

Graphs are general data representations that exist in a wide variety of real-world problems, such as analyzing social networks [1], [2], forecasting traffic flow [3], [4], predicting side-effects or interactions for drug discovery [5]–[7], and recommending products based on personal preferences [8], [9]. Owing to breakthroughs in deep learning, recent graph neural networks (GNNs) [10] have achieved considerable success on diverse graph problems by collectively aggregating information from graph structures [11]–[13]. As a result, in recent years, much research has been focused on how to aggregate feature representations of neighboring nodes to effectively utilize the dependence of the graph.

The associate editor coordinating the review of this manuscript and approving it for publication was Sun-Yuan Hsieh.

The majority of studies have predominantly depended on edges to aggregate the neighboring nodes' information. These edge-based methods are premised on the concept of relational inductive bias within graphs [14], which implies that two connected nodes have similar properties and are more likely to share the same label [15]. Although this approach leverages the unique properties of graphs for structural relations, it appears less capable of generalizing to new or unseen graphs [16].

To improve the neighborhood aggregation scheme, some studies have incorporated node information; they fully utilize node information and reduce the effects of structural (edge) information. A recent approach, graph attention networks (GAT), employs the attention mechanism for adjusting aggregation of neighboring nodes depending on the node information [17]. This approach has yielded impressive

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

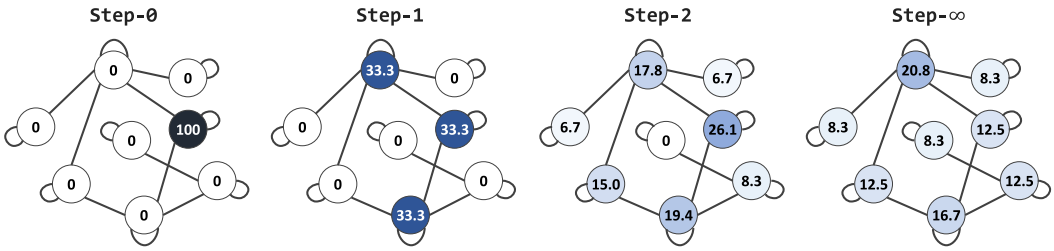


FIGURE 1. Random walk aggregation and propagation procedure. From left to right are step-0, step-1, step-2, and step-infinite. The values in each node indicate the distribution of a random walk. In the leftmost picture, only the starting node has a value of 100, and all other nodes are initialized to zero. As the number of steps increases, values spread further throughout the graph, become smoothed (indistinguishable), and converge to some extent.

performance and has shown potential for improving generalization to unseen graphs.

Regardless of neighborhood aggregation or propagation schemes, most methods, however, suffer from a common problem called oversmoothing. It means that as the aggregation step goes deeper, all node representations become similar and eventually indistinguishable. To avoid the oversmoothing issue, neighbor information is considered within a limited degree [18], weakening the rich graph representation. For example, graph convolutional networks (GCNs) [15] limit the aggregation to shallow depths and only operate on nodes that are closely connected, to keep distinguishable node information [19]. Consequently, information becomes localized and access to global information is restricted [12], leading to poor generalization performance on datasets with only a few labeled nodes [19].

In order to address the aforementioned issues, we propose a novel method, **G**raph **E**ntities with **S**tep **M**ixture via *random walk* (GESM), which considers information from all nodes in the graph and can be generalized to new graphs by incorporating mixture of *random walks* and *attention*. Mixture of *random walks* enriches the graph representation and alleviates the oversmoothing issue, allowing global information to be considered. Hence, our method can be effective particularly on the periphery or a sparsely labeled dataset. The *attention* mechanism also advances our model by adaptively aggregating neighbor node information. This enhances the generalizability of models to diverse graph structures.

Despite of the attention mechanism, it is likely that some neighbor nodes are still not clustered closely on the embedding space. We employ a triplet loss-based regularization technique [20], which enforces the neighbor nodes to be closer and the irrelevant nodes to be distant.

To validate our approach, we conducted experiments on eight standard benchmark datasets: Cora, Citeseer, and Pubmed, which are well-known citation networks datsets, Coauthor CS, Coauthor Physics, Amazon Computers and Amazon Photo, which are co-authorship and co-purchase datsets for transductive learning. We also conducted experiments on protein-protein interaction (PPI) dataset for inductive learning, in which test graphs remain unseen during training. In addition to these experiments, we verified whether our model uses information of remote nodes by reducing the labeled ratio. The experimental results demonstrate the consistently competitive performances of GESM on all the datasets including transductive and inductive scenarios. This manuscript is an expanded version of our prior workshop paper [21].

The key contributions of our approach are as follows:

- We propose **G**raph **E**ntities with **S**tep **M**ixture via *random walk* (GESM), which incorporates **S**tep **M**ixture with bilinear pooling based attention and novel regularization technique.
- We experimentally demonstrate that our proposed model is consistently competitive compared to other models on eight benchmark datasets and is applicable to both transductive and inductive learning tasks. In addition, we show its effectiveness in terms of global aggregation as the rate of labels decreases.
- We provide an in-depth analysis regarding the effects on performance and inference time as the aggregation step increases, and confirm our superiority on the oversmoothing issue.

II. PRELIMINARY

A. RANDOM WALKS

Random walk, a widely used method in graph theory, mathematically models how node information aggregates and propagates throughout the graph. Random walk refers to randomly moving from the starting node to neighbor nodes in a graph in a succession of random steps. FIGURE 1 shows how values change for random walk aggregation and propagation procedure from zero to infinite steps.

For a given graph, the transition matrix $\boldsymbol{P}$, which describes the probabilities of transition, can be formulated as follows:

$$\boldsymbol{P} = \boldsymbol{A}\boldsymbol{D}^{-1} \tag{1}$$

where $\boldsymbol{A}$ denotes the adjacency matrix of the graph, and $\boldsymbol{D}$ the diagonal matrix with a degree of nodes. The probability of moving from one node to any of its neighbors is equal, and the sum of the probabilities of moving to a neighboring node adds up to one.

Let u^t be the distribution of the random walk at step t (u^0 represents the starting distribution). The t step random walk distribution is equal to multiplying $\boldsymbol{P}$, the transition

matrix, t times. In other words,

$$u^1 = \boldsymbol{P}u^0$$
$$u^t = \boldsymbol{P}u^{t-1} = \boldsymbol{P}^t u^0. \quad (2)$$

The entries of the transition matrix are all positive numbers, and each column sums up to one, indicating that $\boldsymbol{P}$ is a matrix form of the Markov chain with steady-state. One of the eigenvalues is equal to 1, and its eigenvector is a steady-state [22]. Therefore, even if the transition matrix is infinitely multiplied, convergence is guaranteed.

B. ATTENTION

The attention mechanism was introduced in sequence-to-sequence modeling to solve long-term dependency problems that occur in machine translation [23]. The key idea of attention is allowing the model to learn and focus on what is important by examining features of the hidden layer. In the case of GNNs [10], attention mechanism has been used to give different importance to neighboring nodes depending on their informational relevance to a center node. During the propagation process, node features are given more emphasis than structural information (edges). Consequently, using attention is advantageous for training and testing graphs with different node features even in the same structures (edges). GAT [17] achieved state-of-the-art performance by using the attention mechanism, which is based on concatenation operation between node information.

Given the many benefits of attention, we incorporate the attention mechanism to our model, which is based on bilinear pooling to fully utilize node information and fine-grained relevance. Our attention mechanism enables different weights to neighboring nodes by considering interactions. Combining attention with mixture-step random walk allows our model to adaptively highlight features with salient information in a global scope.

C. NODE CLASSIFICATION TASK

Node classification is a task to classify the labels of masked nodes by learning from the other nodes of given graphs. Node classification can be classified into *transductive learning* and *inductive learning* tasks based on the usage of unseen graphs.

In *transductive learning* tasks, graph learning is performed within a single fixed graph, and focuses on learning the given structural relationships between nodes of the graph. GCN [15] and HGCN [24], which are state-of-the-art models of transductive learning, focus on how to effectively handle the graph's structural representation. However, these models learn from the spectral-domain or hyperbolic-space by limiting the graph structure to only one fixed graph [25], thus leading to poor generalization on unseen graphs.

In *inductive learning* tasks, graph learning is performed to deal with multiple graphs; it focuses on learning dynamic relationships between nodes with new graphs. Existing models such as GraphSAGE [25] and GAT [17] focused on the local range of aggregations by sampling neighborhood nodes and actively utilizing node embeddings. However, in the case of node sampling, there are issues of which node and how many nodes to sample. Other challenges pertaining to node sampling include the excessive computation time and loss of structural information.

In general, most existing approaches for graph representation learning commonly suffer from the oversmoothing issue. Despite the efforts of recent work including JK-Net [12] and APPNP [18], these methods cannot completely solve the oversmoothing issue. Since APPNP employs a simple sum of Neumann series [26], it cannot adjust to the global and local aggregation scheme. JK-Net similarly does not work effectively for global aggregation as shown in FIGURE 5.

III. METHODOLOGY

Our proposed method enriches graph representation by employing a mixture of various steps through random walk, an attention mechanism for dynamic aggregation, and structure-based embedding regularization. Based on the base component of step mixture, we can add the attention mechanism and regularization.

The notations used in this manuscript are as follows. Let $\mathcal{G} = (V, E)$ be a graph, where V and E denote the sets of nodes and edges, respectively. Nodes are represented as a feature matrix $X \in \mathbb{R}^{n \times f}$, where n and f respectively denote the number of nodes and the input dimension per node. A label matrix is $Y \in \mathbb{R}^{n \times c}$ with the number of classes c, and a learnable weight matrix is denoted by W. The adjacency matrix of graph $\mathcal{G}$ is represented as $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. The addition of self-loops to the adjacency matrix is $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_n$, and the column normalized matrix of $\widetilde{\boldsymbol{A}}$ is $\hat{\widetilde{\boldsymbol{A}}} = \widetilde{\boldsymbol{A}}\boldsymbol{D}^{-1}$ with $\hat{\widetilde{\boldsymbol{A}}}^0 = \boldsymbol{I}_n$.

A. STEP MIXTURE

The step mixture as our base component has a simple structure that is composed of three stages as shown in FIGURE 2. First, input X passes through a fully connected layer with a nonlinear activation and generates embedding node features $Z = \sigma(XW)$. Second, Z is multiplied by a normalized adjacency matrix $\hat{\widetilde{\boldsymbol{A}}}$ for each random walk step that is to be considered. As indicated in the first and second stages, the node embedding and propagation processes are separated. Finally, the step mixture is implemented by concatenating each step and passing f_{concat} through the last prediction layer to produce $\hat{y}$. The entire propagation process can be formulated as:

$$f_{\text{concat}} = \Big\|_{k=0}^{s} \hat{\widetilde{\boldsymbol{A}}}^k \sigma(XW)$$
$$= \Big\|_{k=0}^{s} \hat{\widetilde{\boldsymbol{A}}}^k Z, \quad (3)$$

$$\hat{y} = \text{softmax}((\Big\|_{k=0}^{s} \hat{\widetilde{\boldsymbol{A}}}^k Z)W)), \quad (4)$$

where $\|$ is the concatenation operation, s is the maximum number of steps considered for aggregation, and $\hat{\widetilde{\boldsymbol{A}}}^k$ is the

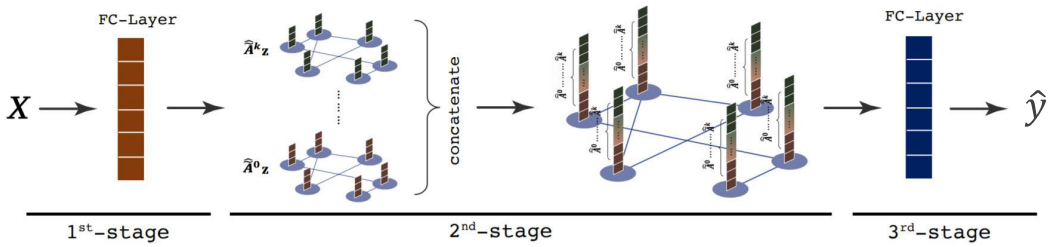


FIGURE 2. Schematic process of our base component: Step Mixture. The procedure consists of three stages: (1st-stage) input passes through the FC layer and produces node feature $Z = \sigma(XW)$, (2nd-stage) Z is multiplied by a adjacency matrix for multiple steps then concatenated, (3rd-stage) the final output is produced.

normalized adjacency matrix $\hat{\tilde{A}}$ multiplied k times. As shown in Equation 3, weights are shared across nodes.

In our method, the adjacency matrix $\hat{\tilde{A}}$ is an asymmetric matrix, which is generated by *random walks* and is flexible to arbitrary graphs. On the other hand, prior methods such as JK-Net [12] and MixHop [27], use a symmetric Laplacian adjacency matrix, which limits graph structures to given fixed graphs.

For step mixture, localized sub-graphs are concatenated with global graphs, which allows the neural network to adaptively select global and local information through learning (see FIGURE 3). While traditional graph convolution methods consider aggregated information within three steps by $A(A(AXW^{(0)})W^{(1)})W^{(2)}$, our method can take all previous aggregations into account by $(A^0Z||A^1Z||A^2Z||A^3Z)$.

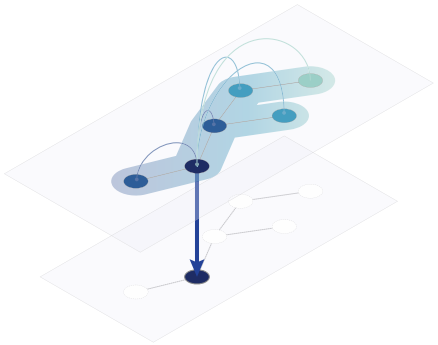
Most graph neural networks suffer from the oversmoothing issue. Although JK-Net [12] tried to handle oversmoothing by utilizing GCN blocks with mulitple propagation, it could not completely resolve the issue as shown in FIGURE 5. Unlike JK-Net, we explicitly separate the node embedding and the propagation process by employing a mixture of multiple random walk steps. This step mixture approach allows our model to alleviate the oversmoothing issue along with localized aggregation.

B. ATTENTION MECHANISM

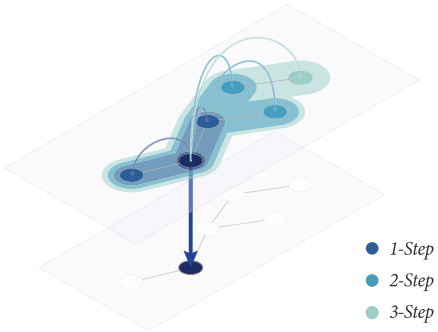
For a more sophisticated design of node embedding Z, we adopt an attention mechanism that dynamically aggregates the information of neighboring nodes according to the relationships between the targets and their neighboring nodes. The degree of relationship, denoted as attention coefficient α, is based on bilinear pooling of encoding vectors between two nodes to reflect all element-level relationships. Bilinear pooling-based attention can consider element-level interactions different from the concatenation-based attention used in GAT.

The attention mechanism is implemented by simply replacing Z in Equation 4 with the attention feature denoted by H_{multi} in Equation 5,

$$\hat{y} = \text{softmax}((\Big\|_{k=0}^{s} \hat{\tilde{A}}^k H_{\text{multi}})W)). \tag{5}$$



(a) Traditional global aggregation scheme



(b) Our step-mixture scheme

FIGURE 3. Conceptual scheme of neighborhood aggregation for three steps (a) in conventional graph neural networks and (b) our method which uses mixture of random walks.

As described in Equation 6 and 7, we employ multi-head attention, where H_{multi} and α_i denote the concatenation of m attention layers and the i-th attention coefficient respectively. Z_i denotes the node embedding of each attention head. We only compute α for targets and their neighboring nodes, to maintain the structure of the graph.

$$H_{\text{multi}} = \Big\|_{i=1}^{m} \sigma(\alpha_i Z_i)), \tag{6}$$

$$Z_i = XW_i, \tag{7}$$

The attention coefficients α is calculated by the sum of outer products between encoding vectors of target node t and its neighbor node j for all neighbor nodes:

$$\alpha = \text{softmax}_j(\sum e_t \otimes e_j), \tag{8}$$

TABLE 1. Summary of datasets used in the experiments.

Type	Datasets	Nodes (Graphs)				Features	Edges	Classes	Label rate
		Total	Training	Validation	Test				
T*	Cora	2,708	140	500	1,000	1,433	5,429	7	5.1%
	Citeseer	3,327	120	500	1,000	3,703	4,732	6	3.6%
	Pubmed	19,717	60	500	1,000	500	44,338	3	0.3%
I†	PPI	56,944 (24)	44,906 (20)	6,514 (2)	5,524 (2)	50	8,187	121	-

* Transductive learning datasets consist of one fixed graph and use a subset of node labels for training.
† Inductive learning datasets consist of many graphs and use some graphs for training and some unseen graphs for test.

where $\otimes$ is an outer product, e is Hadamard product of node embedding Z and weight matrix W, $e = Z \odot W$.

By incorporating attention to our base model, we can avoid or ignore noisy parts of the graph, providing a guide for random walk [28]. Utilizing attention can also improve combinatorial generalization for inductive learning, where training and testing graphs are completely different. In particular, datasets with the same structure but different node information can benefit from our method because these datasets can only be distinguished by node information. Focusing on node features for aggregation can thus provide more reliable results in inductive learning.

C. EMBEDDING REGULARIZATION

Although our attention effectively filters out noisy nodes, there is still a possibility that the correct neighbor nodes may not be clustered in the process, as shown in FIGURE 8. Therefore, it is necessary for node embedding Z to have a regularization that helps cluster neighbor nodes together and move irrelevant nodes further away by reflecting the graph structure. Our push and pull-based triplet regularization R can be formulated as:

$$R = \frac{1}{|S|} \sum_{\substack{p \in S \\ n \in S^c}} \left(\beta \cdot \mathrm{Dis}(Z_c, Z_p) - (1 - \beta) \cdot \mathrm{Dis}(Z_c, Z_n) \right), \quad (9)$$

where $S \subset E$ and its cardinality $|S|$ is the number of samples, and β denotes a weight for the distances between the center node embedding and its positive and negative node embeddings. A positive node p represents the neighbor node of a center node c, and a negative node n represents all the nodes except the positive and center nodes. The distance function, Dis(), uses a sigmoid of dot products, i.e., $\mathrm{Dis}(Z_i, Z_j) = 1 - \mathrm{sigmoid}(Z_i \cdot Z_j)$. The distance between positive nodes and the center are minimized and the distance between negative nodes and the center node are maximized.

Finally, our objective function $\mathcal{L}$ is $\mathcal{L} = J + R$, where J denotes a cross-entropy loss between the prediction label $\hat{y}$ and the target label to reduce classification errors, and R denotes an embedding regularizer to further differentiate embedding representations.

IV. EXPERIMENTS

A. DATASETS

1) TRANSDUCTIVE LEARNING

We utilize three benchmark datasets for node classification: Cora, Citeseer, and Pubmed [34]. These three datasets are citation networks, in which the nodes represent documents and the edges correspond to citation links. The edge configuration is undirected, and the feature of each node consists of word representations of a document. Detailed statistics of the datasets are described in Table 1.

For experiments on datasets with the public label rate, we follow the transductive experimental setup of [35]. Although all of the feature vectors of node are accessible, only 20 node labels per class are used for training. Accordingly, 5.1% for Cora, 3.6% for Citeseer, and 0.3% for Pubmed can be learned. In addition to the experiments with public label rate settings, we conducted the experiments using datasets where labels were randomly split into a smaller set for training. To check whether our model can propagate node information to the entire graph, we reduced the label rate of Cora to 3% and 1%, Citeseer to 1% and 0.5%, Pubmed to 0.1%, and followed the experimental settings of [19] for these datasets with low label rates. For all of the experiments, we report the results using 1,000 test nodes and 500 validation nodes.

2) INDUCTIVE LEARNING

We use the protein-protein interaction PPI dataset [36],which is preprocessed by [17]. As detailed in Table 1, the PPI dataset consists of 24 different graphs, where 20 graphs are used for training, 2 for validation, and 2 for testing. The test set remains completely unobserved during training. Each node is multi-labeled with 121 labels and 50 features regarding gene sets and immunological signatures.

3) ROBUSTNESS

For an in-depth verification of overfitting, we extended our experiments to four types of new node classification datasets. Coauthor CS and Coauthor Physics are co-authorship graphs from the KDD Cup 2016 challenge[2], in which nodes are authors, features represent the article keyword for each

[2]https://kddcup2016.azurewebsites.net/

TABLE 2. Summary of coauthor and Amazon co-purchase datasets used for robustness experiments.

	Classes	Features	Nodes	Edges
Coauthor CS	15	6,805	18,333	100,227
Coauthor Physics	5	8,415	34,493	282,455
Amazon Computers	10	767	13,381	259,159
Amazon Photo	8	745	7,487	126,530

author's paper, and class labels indicate each author's most active research areas. Amazon Computers and Amazon Photo are co-purchase graphs of Amazon, where nodes represent the items, and edges indicate that items have been purchased together. The node features are bag-of-words of product reviews, and class labels represent product categories. Detailed statistics of the datasets are described in Table 2.

B. EXPERIMENTAL SETUP

Regarding the hyperparameters of our transductive learning models, we set the dropout probability such that 0.7 and number of multi-head $m = 8$. The size of the hidden layer $h \in \{64, 512\}$ and the maximum number of steps used for aggregation $s \in \{5, 15\}$ were adjusted for each dataset. We trained for a maximum of 300 epochs with L2 regularization $\lambda \in \{0.003, 0.0005\}$, triplet regularization $\beta \in \{0.5, 1.0\}$ and learning rate $lr \in \{0.003, 0.0008\}$. We report the average classification accuracy of 20 runs.

For inductive learning, we set the size of hidden layer $h = 256$, number of steps $s = 3$, multi-head attention $m = 8$, and $\beta = 1.0$ for GESM. L2 regularization and dropout were not used for inductive learning [17]. We trained our models for a maximum of 3,000 epochs with learning rate $lr = 0.008$. The evaluation metric is the micro-F1 score, and we report the averaged results of 10 runs.

Our models were initialized using Glorot initialization [37], and the nonlinearity function of the first fully connected layer was an exponential linear unit (ELU) [38]. Models were trained to minimize the cross-entropy loss using the Adam optimizer [39]. We employed an early stopping strategy based on the loss and accuracy of the validation sets, 20 epochs for Cora and 100 epochs for others. All the experiments were performed through NAVER Smart Machine Learning (NSML) platform [40], [41].

V. RESULTS

A. NODE CLASSIFICATION

1) RESULTS ON BENCHMARK DATASETS

Table 3 summarizes the experimental results for transductive and inductive learning tasks which were based on the public label ratio. In general, not only are there a small number of methods that can perform on both transductive and inductive learning tasks, but the performance of such methods is not consistently high. Our methods, however, are ranked in the top-3 for every task, indicating that our method can be applied to any task with large predictive power.

For transductive learning tasks, the experimental results of our methods are higher than or equivalent to those of other methods. As can be identified from the table, our model GESM (w/o reg), which considers node information in the aggregation process, outperforms many existing baseline models. These results indicate the significance of considering both global and local information with the attention mechanism. It can also be observed that GESM achieves more stable results than GESM (w/o reg), suggesting the importance of reconstructing node representations. In addition, we calculate confidence intervals via bootstrapping, denoted by GESM (bootstrap.). For the bootstrapping setup, we varied the train nodes with replacement and fixed valid/test nodes. This demonstrates a statistically rigorous performance of GESM.

For the inductive learning task, our proposed model GESM (w/o reg) and GESM outperform the results of GAT, despite the fact that GAT consists of more attention layers. These results for unseen graphs are consistent with those from the previous work [17], that improved generalization by enhancing the influence of node information. JK-Net shows comparable performance to GESM on PPI dataset. However, subsequent in-depth comparisons on the oversmoothing issue confirm the superiority of our method compared to JK-Net.

2) RESULTS ON DATASETS WITH LOW LABEL RATES

To demonstrate that our methods can consider global information, we experimented on transductive learning datasets with low label rates. As indicated in Table 4, our models show remarkable performance even on sparse datasets. In particular, we can observe that our methods that were trained on only 3% of Cora dataset outperformed some methods trained on 5.1% of the data. Since each run of the experiment uses a separate random split with very low label rates (e.g., 1% Cora is only 25 nodes for the train), the overall bootstrapped confidence interval is larger than its public counterpart.

It could be speculated from the results that using a mixture of random walks played a key role in the accuracy of our proposed model GESM; the improved results can be explained by our method adaptively selecting node information from local and global neighborhoods, and allowing even peripheral nodes to receive information.

3) EXPERIMENTS ON OTHER DATASETS FOR ROBUSTNESS

Existing works have suffered from experimental bias and overfitting from using a single train/valid/test split, and tuning hyperparameters to each dataset. GNN algorithms are very sensitive to both data splits and weight initialization (as shown by [18], [42]). Thus, [42] suggests carefully designed evaluation protocol which runs experiments on multiple random splits and initializations.

To check robustness, we extended our experiments to Coauthor CS, Coauthor Physics, Amazon Computers, and Amazon Photo datasets. We followed the experimental

TABLE 3. Experimental results on the public benchmark datasets. Evaluation metrics on transductive and inductive learning datasets are classification accuracy (%) and F1-score, respectively. We report the uncertainties of GESM (bootstrap.) showing the 95% confidence level calculated by bootstrapping. Top-3 results for each column are highlighted in bold, and top-1 values are underlined.

	Transductive			Inductive
Method	Cora public (5.1%)	Citeseer public (3.6%)	Pubmed public (0.3%)	PPI
Cheby [29]	81.2	69.8	74.4	-
GCN [15]	81.5	70.3	79.0	-
GraphSAGE [25]	-	-	-	0.612
GAT [17]	83.0	**72.5**	79.0	0.973
JK-Net [12]	-	-	-	**0.976**
AGNN [30]	83.1	71.7	79.9	-
Union [19]	80.5	65.7	78.3	-
APPNP [18]*	**83.2**	71.7	79.7	-
SGC [31]	81.0	71.9	78.9	-
MixHop [27]	81.9	71.4	**80.8**	-
GWNN [32]	82.8	71.7	79.1	-
AdaLNet [33]	80.4	68.7	78.1	-
HGCN [24]	79.9	-	**80.3**	-
GESM (w/o att, reg)†	82.8	71.7	**80.3**	0.753
GESM (w/o reg)‡	**84.4**	**72.6**	80.1	**0.976**
GESM	**84.5**	**72.7**	**80.4**	**0.974**
GESM (bootstrap.)	84.4 ± 0.109	72.5 ± 0.180	80.7 ± 0.110	-

* Best experimental results through our own implementation
† GESM (w/o att, reg) for GESM without attention mechanism and regularization
‡ GESM (w/o reg) for GESM without regularization

TABLE 4. Node classification results on datasets with low label rates. Top-3 results for each column are highlighted in bold and top-1 values are underlined. The uncertainties of GESM showing the 95% confidence level calculated by bootstrapping.

	Cora		Citeseer		Pubmed
Method	1%	3%	0.5%	1%	0.1%
Cheby [29]	52.0	70.8	31.7	42.8	51.2
GCN [15]	62.3	76.5	43.6	55.3	65.9
Union [19]	**69.9**	78.5	46.3	59.1	70.7
APPNP [18]*	67.6	**80.8**	40.5	59.9	70.7
SGC [31]*	64.2	77.2	41.0	58.1	71.7
AdaLNet [33]	67.5	77.7	**53.8**	**63.3**	**72.8**
GESM (w/o att, reg)	68.2 ± 3.51	**81.6** ± 1.40	45.6 ± 1.93	62.6 ± 1.27	**73.0** ± 1.31
GESM (w/o reg)	**70.5** ± 2.89	**81.2** ± 1.05	**53.2** ± 1.62	**62.7** ± 0.964	**73.8** ± 1.45
GESM	**70.9** ± 3.07	**80.8** ± 1.23	**51.8** ± 1.91	**63.0** ± 1.01	**72.8** ± 1.36

* Best experimental results through our own implementation

setup of [42]. The experiment is performed by repeating 100 random train/valid/test splits with 20 random weight initializations and using the same hyperparameters (unified size: 64, step size: 15, multi-head for GAT and GESM: 8) without additional tuning. Table 5 proves that our proposed methods do not overfit to any particular dataset and consistently ranks in the top-3 methods. Moreover, in comparison to GAT, GAT shows a very high standard deviation on Amazon Computers and Amazon Photo, whereas the performance of GESM is more accurate and stable. For a more rigorous study of statistical validation, we report the uncertainties of models showing the 95% confidence level (see Table 6).

B. MODEL ANALYSIS

1) OVERSMOOTHING AND ACCURACY

To investigate the oversmoothing issue empirically, we conducted an experiment to check the classification accuracy as the number of aggregation or propagation steps increased. As shown in FIGURE 4, GCN [15], SGC [31], and GAT [17] suffer from oversmoothing. GCN and GAT show severe degradation in accuracy after the 8th step; The accuracy of SGC does not drop as much as GCN and GAT but nevertheless gradually decreases as the step size increases. Unlike the others, our proposed GESM, maintains its performance without any degradation due to no rank loss [43] and alleviating the oversmoothing by step mixture.

TABLE 5. Average test set accuracy and standard deviation over 100 random train/validation/test splits with 20 runs. Top-3 results for each column are highlighted in bold, and top-1 values are underlined.

	Coauthor CS	Coauthor Physics	Amazon Computers	Amazon Photo
MLP	88.3 ± 0.7	88.9 ± 1.1	44.9 ± 5.8	69.6 ± 3.8
LogReg	86.4 ± 0.9	86.7 ± 1.5	64.1 ± 5.7	73.0 ± 6.5
GCN [15]	91.1 ± 0.5	92.8 ± 1.0	**82.6** ± 2.4	**91.2** ± 1.2
GraphSAGE [25]	91.3 ± 2.8	93.0 ± 0.8	**82.4** ± 1.8	**91.4** ± 1.3
GAT [17]	90.5 ± 0.6	92.5 ± 0.9	78.0 ± 19.0	85.7 ± 20.3
GESM (w/o att, reg)	**91.8** ± 0.4	**93.3** ± 0.6	79.2 ± 2.0	89.3 ± 1.9
GESM (w/o reg)	**91.5** ± 0.5	**93.4** ± 0.6	80.6 ± 2.1	89.8 ± 1.9
GESM	**91.4** ± 0.5	**93.5** ± 0.8	**80.8** ± 2.0	**90.3** ± 2.1

TABLE 6. Average test set accuracy with uncertainties showing the 95% confidence level over 100 random train/validation/test splits with 20 runs.

	Coauthor CS	Coauthor Physics	Amazon Computers	Amazon Photo
MLP	88.3 ± 0.030	88.9 ± 0.048	44.9 ± 0.254	69.6 ± 0.167
LogReg	86.4 ± 0.039	86.7 ± 0.065	64.1 ± 0.250	73.0 ± 0.285
GCN [15]	91.1 ± 0.022	92.8 ± 0.044	**82.6** ± 0.105	**91.2** ± 0.053
GraphSAGE [25]	91.3 ± 0.123	93.0 ± 0.035	**82.4** ± 0.079	**91.4** ± 0.057
GAT [17]	90.5 ± 0.026	92.5 ± 0.039	78.0 ± 0.833	85.7 ± 0.890
GESM (w/o att, reg)	**91.8** ± 0.018	**93.3** ± 0.026	79.2 ± 0.088	89.3 ± 0.083
GESM (w/o reg)	**91.5** ± 0.022	**93.4** ± 0.026	80.6 ± 0.092	89.8 ± 0.083
GESM	**91.4** ± 0.022	**93.5** ± 0.035	**80.8** ± 0.088	**90.3** ± 0.092

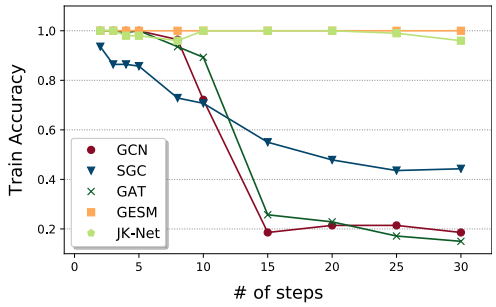


FIGURE 4. Training accuracy comparison according to the number of aggregation (or propagation) steps.

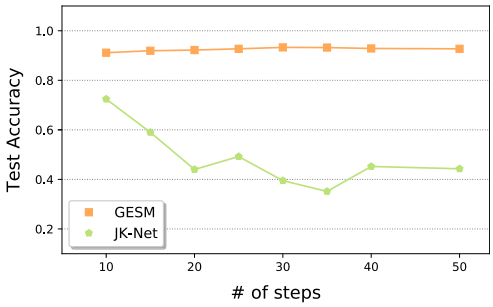


FIGURE 5. Test accuracy of JK-Net and GESM using concatenated features after the 10th step.

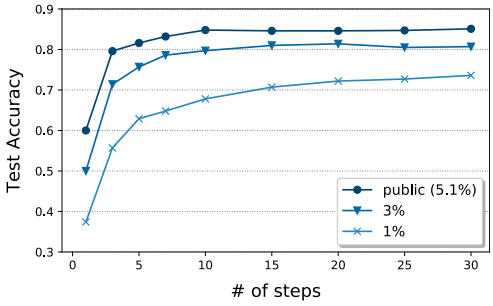


FIGURE 6. GESM Accuracy comparison for various label rates according to step size.

Intriguingly, JK-Net [12] also maintains the training accuracy regardless of the step size by using GCN blocks with multiple steps according to FIGURE 4. Like a similar approach on earlier, we further compared the test accuracy of GESM with JK-Net, by the step size. To inspect the adaptability to larger steps of GESM and JK-Net, we concatenated features after the 10-th step. As shown in FIGURE 5, GESM outperforms JK-Net, even though both methods use the concatenated features. These results demonstrated that JK-Net obtains global information similar to GCN or GAT. Consequently, the larger the step, the more difficult it is for JK-Net to maintain test accuracy. GESM, on the other hand, maintains a steady performance, which confirms that GESM does not collapse even in large steps.

We also observe the effect on accuracy as the number of steps increases under various label rates for GESM. As represented in FIGURE 6, it is evident that considering remote nodes can contribute to the increase in accuracy. By taking into account more data within a larger neighborhood, our model can make reliable decisions. The figure also indicates that the accuracy converges faster for datasets with higher

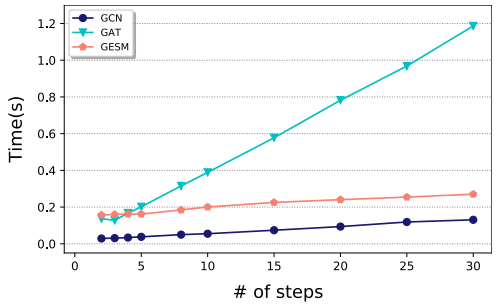
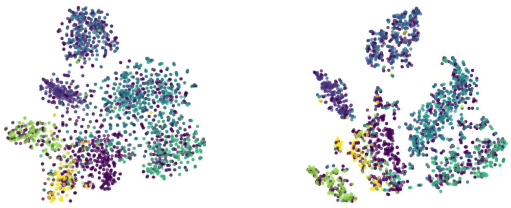


FIGURE 7. Inference time of various models as the step size increases on Cora.



(a) GESM (w/o reg) (b) GESM

FIGURE 8. t-SNE plot of the last hidden representation trained on Cora depending on the existence of regularizer.

label rates, presumably because a small number of walk steps can be used to explore the entire graph.

2) INFERENCE TIME

Computational complexity of all methods increases linearly as the step size increases, as shown in FIGURE 7. We can observe that the inference time of GCN [15] is slightly faster than that of GESM, especially with a constant margin. The inference time of GESM is much faster than GAT [17], even though both methods employ multi-head attention mechanisms. GESM provides higher accuracy in a reasonable inference time due to the sophisticated design comprising a mixture of random walk steps.

3) EFFECTS OF REGULARIZER

The last embedding representations of the proposed methods are visualized using the t-SNE [44], as represented in FIGURE 8. The figure illustrates the difference between GESM (w/o reg) and GESM (with reg). While the nodes are scattered in GESM (w/o reg), they are closely clustered in that of GESM due to the effects of regularization. According to the results in TABLE 3, more distinctly clustered GESM generally produce better results than loosely clustered GESM without regularization.

VI. CONCLUSION

Conventional graph neural networks suffer from the over-smoothing issue as the number of aggregation (or propagation) steps increase and from poor generalization on unseen graphs. To tackle these issues, we propose a simple but effective model that weights differently depending on node information in the aggregation process and adaptively considers global and local information by employing the mixture of multiple random walk steps. To further refine the graph representation, we have presented a new regularization term, which enforces the similar neighbor nodes to be closely clustered in the node embedding space. The results from extensive experiments show that our GESM successfully achieves state-of-the-art or competitive performance for both transductive and inductive learning tasks on eight benchmark graph datasets.

For future research, we will refine our method that utilizes node information to improve the computational efficiency for attention. In addition, we will extend GESM so that it can be applied to real-world large-scale graph data.

REFERENCES

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.
[2] J. Jia, B. Wang, and N. Z. Gong, "Random walk based fake account detection in online social networks," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2017, pp. 273–284.
[3] E. Manley, "Estimating urban traffic patterns through probabilistic interconnectivity of road network junctions," *PLoS ONE*, vol. 10, no. 5, May 2015, Art. no. e0127095.
[4] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," 2017, *arXiv:1709.04875*. [Online]. Available: http://arxiv.org/abs/1709.04875
[5] W. Torng and R. B. Altman, "Graph convolutional neural networks for predicting drug-target interactions," *J. Chem. Inf. Model.*, vol. 59, no. 10, pp. 4131–4149, Oct. 2019.
[6] J. Lim, S. Ryu, K. Park, Y. J. Choe, J. Ham, and W. Y. Kim, "Predicting drug–target interaction using a novel graph neural network with 3D structure-embedded graph representation," *J. Chem. Inf. Model.*, vol. 59, no. 9, pp. 3981–3988, Sep. 2019.
[7] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, Jul. 2018.
[8] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford InfoLab, Stanford, CA, USA, Tech. Rep., 1999.
[9] K.-M. Kim, D. Kwak, H. Kwak, Y.-J. Park, S. Sim, J.-H. Cho, M. Kim, J. Kwon, N. Sung, and J.-W. Ha, "Tripartite heterogeneous graph propagation for large-scale social recommendation," 2019, *arXiv:1908.02569*. [Online]. Available: http://arxiv.org/abs/1908.02569
[10] 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, Jan. 2009.
[11] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learning with generative adversarial nets," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[12] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1–8.
[13] H. Gao and S. Ji, "Graph representation learning via hard and channel-wise attention networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 741–749.
[14] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*. [Online]. Available: http://arxiv.org/abs/1806.01261
[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–14.
[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019, *arXiv:1901.00596*. [Online]. Available: http://arxiv.org/abs/1901.00596
[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–12.

[18] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized PageRank," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–15.
[19] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.
[20] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *Int. J. Comput. Vis.*, vol. 124, no. 2, pp. 237–254, 2017.
[21] K. Shin, W. Shin, J.-W. Ha, and S. Kwon, "Graphs, entities, and step mixture," in *Proc. Int. Conf. Mach. Learn. Workshop Graph Represent. Learn. Beyond*, 2020.
[22] G. Strang, *Introduction to Linear Algebra*. Wellesley, MA, USA: Wellesley-Cambridge Press, 1993, vol. 3.
[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.
[24] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 4869–4880.
[25] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[26] D. F. Gleich and K. Kloster, "Seeded PageRank solution paths," *Eur. J. Appl. Math.*, vol. 27, no. 6, pp. 812–845, Dec. 2016.
[27] S. Abu-El-Haija, B. Perozzi, A. Kapoor, H. Harutyunyan, N. Alipourfard, K. Lerman, G. V. Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 21–29.
[28] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh, "Attention models in graphs: A survey," 2018, *arXiv:1807.07984*. [Online]. Available: http://arxiv.org/abs/1807.07984
[29] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[30] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," 2018, *arXiv:1803.03735*. [Online]. Available: http://arxiv.org/abs/1803.03735
[31] F. Wu, T. Zhang, A. H. D. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6861–6871.
[32] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–13.
[33] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "LanczosNet: Multi-scale deep graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–9.
[34] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, 2008.
[35] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 40–48.
[36] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, Jul. 2017.
[37] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[38] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.
[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.
[40] H. Kim, M. Kim, D. Seo, J. Kim, H. Park, S. Park, H. Jo, K. Kim, Y. Yang, Y. Kim, N. Sung, and J.-W. Ha, "NSML: Meet the MLaaS platform with a real-world case study," 2018, *arXiv:1810.09957*. [Online]. Available: http://arxiv.org/abs/1810.09957
[41] N. Sung, M. Kim, H. Jo, Y. Yang, J. Kim, L. Lausen, Y. Kim, G. Lee, D. Kwak, J.-W. Ha, and S. Kim, "NSML: A machine learning platform that enables you to focus on your models," 2017, *arXiv:1712.05902*. [Online]. Available: http://arxiv.org/abs/1712.05902
[42] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2018, *arXiv:1811.05868*. [Online]. Available: http://arxiv.org/abs/1811.05868
[43] S. Luan, M. Zhao, X.-W. Chang, and D. Precup, "Break the ceiling: Stronger multi-scale deep graph convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 10945–10955.
[44] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.



KYUYONG SHIN received the Bachelor of Science (B.Sc.) degree in field of urban design and planning from Hongik University, South Korea, in 2020. He is currently a Machine Learning Engineer at NAVER AI Laboratory. His research interests include graph neural net, geometric deep learning, and large-scale machine learning. Currently, he has focused on general-purpose representation learning through large-scale pre-training.



WONYOUNG SHIN received the M.S. degree in industrial engineering from KAIST. She is currently a Machine Learning Research Engineer at Naver Shopping. She has years of experience in analyzing data and developing deep learning models for commerce systems, with interests in addressing real-world challenges that hinder the practical use of ML methods.



JUNG-WOO HA (Member, IEEE) received the B.S. and Ph.D. degrees from the Department of Computer Science, Seoul National University. He is currently the Head of NAVER AI Laboratory. He is also working as the Co-Director of two AI research centers cooperating with academia, including SNU-NAVER Hyperscale AI Center and KAIST-NAVER Hypercreative AI Center. He has contributed to many machine learning, computer vision, and natural language processing research communities as the program committees or reviewers, including ICLR, NeurIPS, ICML, CVPR, ICCV, ECCV, ACL, EMNLP, and AAAI. His interests include deep learning-based computer vision, natural language processing, recommendation, and their industrial applications.



SUNYOUNG KWON (Member, IEEE) received the B.S. degree in computer science from Pusan National University, South Korea, in 2001, the M.S. degree in bioinformatics and the Ph.D. degree in electrical and computer engineering from Seoul National University, South Korea, in 2014 and 2018, respectively. From 2018 to 2020, she was with NAVER Corporation, South Korea. Previously, she was with Korea Communication Agency and LG Electronics Inc., South Korea. She is currently an Assistant Professor with the School of Biomedical Convergence Engineering, Pusan National University. Her research interests include AI, bioinformatics, drug discovery, graph neural networks, machine learning, and big-data analytics.