Transactions on Machine Learning and Artificial Intelligence - Vol. 10, No. 3
Publication Date: June, 25, 2022
DOI:10.14738/tmlai.103.12399.
Wu, N., & Wang, C. (2022). Ensemble Graph Attention Networks. Transactions on Machine Learning and Artificial Intelligence, 10(3). 29-41.

Ensemble Graph Attention Networks

Nan Wu
Institute for Financial Services Analytics
University of Delaware, Newark, DE, United States

Chaofan Wang
San Jose, CA, United States

ABSTRACT

Graph neural networks have demonstrated its success in many applications on graph-structured data. Many efforts have been devoted to elaborating new network architectures and learning algorithms over the past decade. The exploration of applying ensemble learning techniques to enhance existing graph algorithms have been overlooked. In this work, we propose a simple generic bagging-based ensemble learning strategy which is applicable to any backbone graph models. We then propose two ensemble graph neural network models – Ensemble-GAT and Ensemble-HetGAT by applying the ensemble strategy to the graph attention network (GAT), and a heterogeneous graph attention network (HetGAT). We demonstrate the effectiveness of the proposed ensemble strategy on GAT and HetGAT through comprehensive experiments with four real-world homogeneous graph datasets and three real-world heterogeneous graph datasets on node classification tasks. The proposed Ensemble-GAT and Ensemble-HetGAT outperform the state-of-the-art graph neural network and heterogeneous graph neural network models on most of the benchmark datasets. The proposed ensemble strategy also alleviates the over-smoothing problem in GAT and HetGAT.

Keywords: Ensemble learning; Bagging; Graph Neural Network; Heterogeneous Graph Neural Network; Graph Attention Network.

INTRODUCTION

Ensemble learning is a general technique to enhance the model performance by combining the predictions from an ensemble of base learners. An ensemble system mainly includes three strategies [1]: (1) strategy of training sample selection for base learners; (2) strategy of training base learners; (3) strategy of fusing decisions from all base learners. There has been an explosion of ensemble-based algorithms in the past decades. Representative prominent works include bagging [2], boosting [3, 4], stacked generalization [5], and mixture of experts (MoE) [6, 7].

Bagging uses independent base learners, and the training samples for each base learner are sampled with replacement (i.e., bootstrapped samples) from the original training set. The decision of Bagging is obtained by majority voting for classification tasks and mean for regression tasks. It is one of the earliest ensemble-based algorithms. Random Forest [8] is an ensemble of decision trees trained with the bagging mechanism and random subspace method

[9]. The training samples for each base decision tree are selected with bootstrapping, and a random subset of the features is selected for each candidate split in the training process.

Boosting [4] is an additive model learning method which sequentially adds a base model to improve the prediction by the prior models. AdaBoost [3] is one of the earliest realizations of the boosting algorithms. It trains base classifiers sequentially based on weighted training samples, and each subsequent classifier focuses more on the samples misclassified by the previous classifier. The decision of AdaBoost is obtained by weighted majority voting, where the weights depend on the training errors of the base classifiers. Compared to AdaBoost, Gradient Boosting [10] is a generic boosting algorithm that works on any differentiable loss function. Gradient boosting decision tree (GBDT) minimizes the loss function by adding weak learners using gradient descent. It generally outperforms random forests.

Instead of none trainable combination of base learners, stacked generalization and MoE use a learning model to combine the output of base learners. The base learners may be of different models. MoE integrates the base learners through a weighted combination, where the weights are learned through a gating network.

Despite the success of Graph Neural Networks (GNNs) in representation learning on graph-structured data, most efforts have been devoted to elaborating new network architectures and learning algorithms, but overlooked the leverage of ensemble learning techniques to improve existing graph algorithms. [11] exploits AdaBoost to combine convolutional information from different orders of neighbors in a graph, and can be viewed as an attempt of ensemble graph learning. Despite the novelty of applying AdaBoost to GCN, this mechanism is computationally challenging to be applied to attention-based GNN models due to the computation requirement for attention weights of each node and its neighbors of arbitrary orders. Similar to TAGCN [12] and DAGNN [13], AdaGCN is a variant of multi-hop graph convolution networks, where the convolution of different orders of neighbors (i.e. $\boldsymbol{A}^l\boldsymbol{X}$) is computed explicitly in the forward network instead of stacking multiple graph convolution layers. Therefore, to apply such multi-hop mechanism to attention-based GNNs, one will need to compute the attention weights between each node and its l-hop neighbors for the l^{th} base learner, where the number of attention weights equals to the non-zero entries of the l^{th} power of the adjacency matrix (i.e., $\boldsymbol{A}^l$). Although the adjacency matrix $\boldsymbol{A}$ is usually a sparse matrix, its l^{th} power could be very dense. For example, the adjacency matrix of the Cora dataset [14] has only about 10k non-zero entries though, its fifth-power has about 2.2 million non-zero entries. Furthermore, to our best knowledge, none of the existing works has proposed to apply ensemble techniques to enhance the performance of heterogeneous graph neural networks (HGNNs).

To fill this gap, we propose a simple yet effective ensemble learning strategy based on a variant of bagging, and apply it to the graph attention network (GAT) [15], named as Ensemble-GAT. We also propose a heterogeneous graph attention work (HetGAT) which is built on GAT by introducing node-type- and edge-type-dependent parameters to encode the heterogeneity in node contents and graph structure, and then apply the proposed ensemble strategy to it, named as Ensemble-HetGAT. For a homogeneous graph, we randomly remove a certain rate of edges to generate an ensemble of input graph samples for each base GAT. All base GATs are then trained in parallel, and the results are combined by majority voting for node classification tasks.

For a heterogeneous graph, we randomly remove a certain rate of edges in each edge type to generate an ensemble of input graph samples for each base HetGAT. The rest processes are the same as for a homogeneous graph. We conduct comprehensive experiments with four real-world homogeneous graph datasets and three real-world heterogeneous graph datasets on the node classification task, and demonstrate the effectiveness of this simple ensemble strategy on improving the performance and robustness of the vanilla GAT and HetGAT. The proposed Ensemble-GAT outperforms the state-of-the-art APPNP [16] and DAGNN [13] models on most of the benchmark datasets. The proposed Ensemble-HetGAT outperforms the state-of-the-art SimpleHGN [17] model on all three benchmark datasets.

PRELIMINARY

In this section, we introduce the definition of graphs and notations used throughout this paper. We then review the vanilla GAT [15] and introduce the proposed HetGAT.

Definition of Graph

A generic graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}^v, \mathcal{T}^e)$ with a node type mapping function $\phi: \mathcal{V} \to \mathcal{T}^v$, and an edge type mapping function $\psi: \mathcal{E} \to \mathcal{T}^e$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{T}^v$ is the set of node types, and $\mathcal{T}^e$ is the set of edge types. $|\mathcal{V}|$ denotes the number of nodes, $|\mathcal{E}|$ denotes the number of edges, $|\mathcal{T}^v|$ denotes the number of node types, and $|\mathcal{T}^e|$ denotes the number of edge types. $e_{u,v} = (u, v) \in \mathcal{E}$ denotes an edge from node v to node u.

When $|\mathcal{T}^v| = |\mathcal{T}^e| = 1$, the graph is a *homogeneous graph*. A homogeneous graph can be fully described by its adjacency matrix $\boldsymbol{A}$ and the corresponding degree matrix $\boldsymbol{D}$. $\mathcal{N}_u$ denotes the set of node u's neighbors. $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ represents the nodes' feature map, where each row $\boldsymbol{x}_u \in \mathbb{R}^{1 \times D}$ represents the feature vector of a node u with the feature dimension of D.

When $|\mathcal{T}^v| + |\mathcal{T}^e| > 2$, the graph is a *heterogeneous graph*. A heterogeneous graph can be fully described by a set of adjacency matrices $\{\boldsymbol{A}_k\}_{k=1}^{|\mathcal{T}^e|}$ and the corresponding set of degree matrices $\{\boldsymbol{D}_k\}_{k=1}^{|\mathcal{T}^e|}$, where $\boldsymbol{A}_k$ is the adjacency matrix of edge type k. $\mathcal{N}_u^k$ denotes the set of node u's neighbors with type k connection. $\mathcal{V}_a$ denotes the set of nodes of type a. Each node $u \in \mathcal{V}_a$ is associated with an input feature vector $\boldsymbol{x}_u \in \mathbb{R}^{1 \times D_a}$ with the feature dimension of D_a. The input feature map of all type a nodes is denoted as $\boldsymbol{X}_{\boldsymbol{a}} \in \mathbb{R}^{|\mathcal{V}_a| \times D_a}$.

Graph Attention Network (GAT)

GAT [15] is one of the earliest attention-based GNNs operating on the spatial domain of a graph. It aggregates information from a node's neighbors with attention-based weights. The feed forward propagation of an L-layer GAT takes the following form:

$$\begin{aligned} \boldsymbol{H}^0 &= \boldsymbol{X} \\ \boldsymbol{h}_u^{l+1} &= \mathrm{ELU}\left(\sum\nolimits_{v \in \mathcal{N}_u \cup \{u\}} \alpha_{u,v}^l \boldsymbol{h}_v^l \boldsymbol{W}^l\right) \\ \alpha_{u,v}^l &= \mathrm{softmax}_v\Big(\mathrm{LeakyReLU}([\boldsymbol{h}_u^l \boldsymbol{W}^l \| \boldsymbol{h}_v^l \boldsymbol{W}^l] \boldsymbol{a}^l)\Big) \end{aligned} \tag{1}$$

where $l = 0, 1, \ldots, L-1$, denoting the layer number. $\boldsymbol{h}_u^l$ denotes the hidden feature vector of node u at the l^{th} propagation layer. $\alpha_{u,v}^l$ is the attention weight of a node u on its neighbor node

URL: http://dx.doi.org/10.14738/tmlai.103.12399

v. $\boldsymbol{a}^l$ is a layer-wise learnable attention vector. $\boldsymbol{W}^l$ is a layer-wise learnable transformation matrix.

Heterogeneous Graph Attention Network (HetGAT)

We propose HetGAT by introducing node-type- and edge-type-dependent parameters into GAT [15], and extend the vanilla GAT to heterogeneous graphs. The proposed HetGAT is consisted of three modules: (1) a node-type-specific feature transformation which projects features of different node types into the same vector space; (2) an edge-type-specific GAT layer which aggregates information from neighbors with the same type of relations; (3) a mean aggregator which combines the information from different relations. The mathematical description of an L-layer HetGAT is formulated as:

$$\begin{aligned} \boldsymbol{h}_u^0 &= \sigma\left(\boldsymbol{x}_u \boldsymbol{W}_{0,a} + \boldsymbol{b}_{0,a}\right) \\ \boldsymbol{h}_{u,k}^l &= \mathrm{GAT}_k\left(\boldsymbol{h}_u^{l-1}, \{\boldsymbol{h}_v^{l-1}, \forall v \in \mathcal{N}_u^k\}\right) \\ \boldsymbol{h}_u^l &= \mathrm{ReLU}\left(\frac{1}{K}\sum_{k=1}^{|\mathcal{T}^e|} \boldsymbol{h}_{u,k}^l\right) \end{aligned} \tag{2}$$

where $l = 1, 2, \ldots, L$, denoting the layer number. $k \in \mathcal{T}^e$ denotes the edge type. $\boldsymbol{h}_{u,k}^l$ denotes the edge-type-specific representation of node u at the l^{th} propagation layer. $\boldsymbol{h}_u^l$ denotes the representation of node u at the l^{th} propagation layer. $\mathrm{GAT}_k(\cdot)$ denotes an edge-type-specific GAT layer with the propagation rule described in Equation (1).

Our Method: Ensemble Graph Attention Networks

In this section, we will introduce the proposed ensemble strategy for GAT and HetGAT. The overall architecture of an ensemble graph neural network is illustrated in Figure 1.

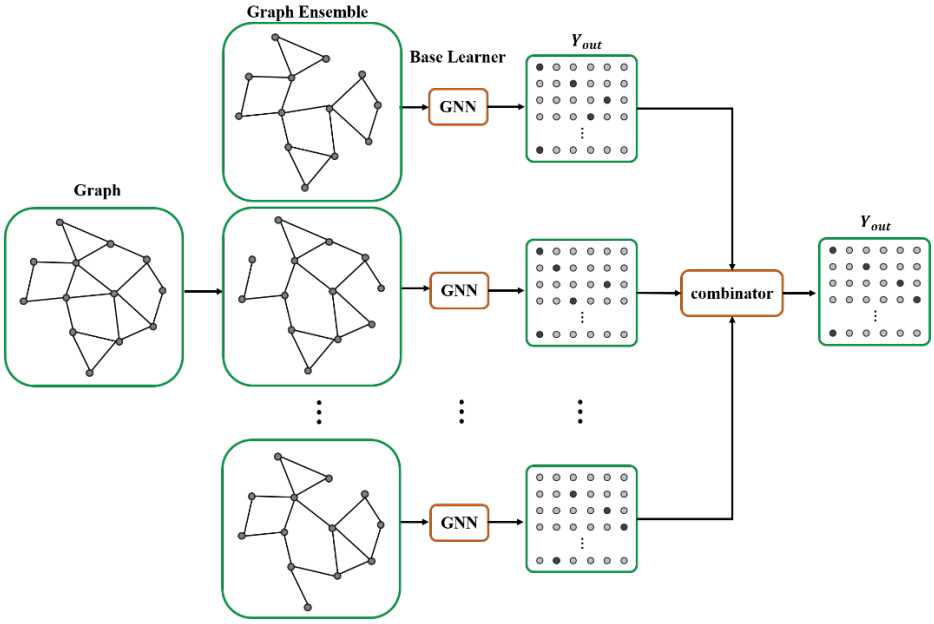


Figure 1 The overall architecture of an ensemble graph neural network

Ensemble-GAT

Given a homogeneous graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and an edge dropping rate $r \in (0,1)$, we first randomly remove $|\mathcal{E}|r$ edges in $\mathcal{G}$ following an identical Bernoulli dropping distribution Bernoulli(r) for each edge $e \in \mathcal{E}$, and repeat the process P times to create a set of perturbed graph samples, $\{\mathcal{G}_i\}_{i=1}^{P}$. We then train a GAT on each perturbed graph sample $\mathcal{G}_i$ for a given downstream stream task. The predictions of each base GAT are then combined through majority voting.

Ensemble-HetGAT

Given a heterogeneous graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T}^v, \mathcal{T}^e)$ and an edge dropping rate $r \in (0,1)$, we first randomly remove $|\mathcal{E}_k|r$ edges for each edge type k in $\mathcal{G}$ following an identical Bernoulli dropping distribution Bernoulli(r) for each edge $e \in \mathcal{E}_k$, where $\mathcal{E}_k$ denotes the set of edges of type $k \in \mathcal{T}^e$. We repeat the process P times to create a set of perturbed graph samples, $\{\mathcal{G}_i\}_{i=1}^{P}$. We then train a HetGAT on each perturbed graph sample $\mathcal{G}_i$ for a given downstream stream task. The predictions of each base HetGAT are then combined through majority voting.

Experiments

Our experiments are conducted in three parts. The first part in Section 0 is to demonstrate the performance of the proposed Ensemble-GAT and Ensemble-HetGAT. The second part in Section 0 is to experiment on the effect of different edge dropping rates and number of ensembles on the performance of Ensemble-GAT and Ensemble-HetGAT. The third part in Section 0 is to examine the improvement of the ensemble strategy on the over-smoothing issue of GAT and HetGAT.

Datasets

We use four popular homogeneous benchmark datasets – Cora, Citeseer, PubMed, and MS Coauthor-CS from the DGL library [14], and three popular heterogeneous benchmark datasets – ACM[1], IMDB[2], and DBLP[3]. Among the four homogeneous graphs, Cora, Citeseer, and PubMed are citation network datasets, and Coauthor-CS is a co-authorship network dataset in the area of Computer Science extracted from the Microsoft Academic Graph. Among the three heterogeneous graphs, ACM is a citation network covering papers in KDD, SIGMOD, VLDB, SIGCOMM, and MOBICOMM, IMDB is a movie network, and DBLP is computer science bibliography network. The statistics of the benchmark datasets are summarized in Table 1 and Table 2.

[1] https://dl.acm.org/
[2] https://www.imdb.com/
[3] https://dblp.uni-trier.de/

URL: http://dx.doi.org/10.14738/tmlai.103.12399

Table 1 Statistics of homogeneous graph datasets

	Cora	Citeseer	PubMed	Coauthor-CS
# Nodes	2,708	3,327	19,717	18,333
# Edges	10,556	9,228	88,651	81,894
# Features/Node	1,433	3,703	500	6,805
# Classes	7	6	3	15
# Training Nodes	20/class	20/class	20/class	20/class
# Validation Nodes	500	500	500	30/class
# Test Nodes	1,000	1,000	1,000	17,583
Label Rate	5.2%	3.6%	0.3%	1.6%

Table 2 Statistics of heterogeneous graph datasets

	ACM	IMDB	DBLP
# Node	Paper (P): 4,019	Movie (M): 4,278	Author (A): 4,057
	Author (A): 7,167	Director (D): 2,081	Paper (P): 14,328
	Subject (S): 60	Actor (A): 5,257	Term (T): 7,723
			Conference (C): 20
# Edge	P-A: 13,407	M-D: 4,278	A-P: 19,645
	P-S: 4,019	M-A: 12,828	P-T: 85,810
			P-C: 14,328
# Node Feature	P: 1,902	M: 3,066	A: 334
	A: 1,902	D: 3,066	P: 334
	S: 1,902	A: 3,066	T: 50
			C: 334
Target Node	Paper	Movie	Author
# Class	3	3	4
# Training	600	300	800
# Validation	300	300	400
# Test	3,119	3,678	2,857

We use fixed data splits for the Cora, Citeseer and PubMed datasets which are provided by the DGL library. For Coauthor-CS, we randomly split the dataset with 20 training samples per class (300 total training samples), 30 validation samples per class (450 total validation samples), and all remaining samples for test. For ACM, we choose 600 balanced samples as the training set, another 300 balanced samples as the validation set and all remaining 3,119 samples as the test set. For IMDB, we choose 300 balanced samples as the training set, another 300 balanced samples as the validation set and all remaining 3,678 samples as the test set. For DBLP, we choose 800 balanced samples as the training set, another 400 balanced samples as the validation set and all remaining 2,857 samples as the test set.

Performance of Ensemble-GAT and Ensemble-HetGAT

In this section, we evaluate the model performance of Ensemble-GAT on four homogeneous benchmark datasets described in

Table 1 on the node classification task, and compare the results with two state-of-the-art homogeneous GNNs – APPNP [16] and DAGNN [13]. We then evaluate the model performance of Ensemble-HetGAT on three heterogeneous benchmark datasets described in
Table 2 on the node classification task, and compare the results with one of the most popular HGNNs – HAN [18] and one state-of-the-art HGNN – SimpleHGN [17].

Experimental Setup

Edge Dropping Rate. We use $r = 20\%$ edge dropping rate on Cora, Citeseer, and PubMed, and $r = 50\%$ edge dropping rate on Coauthor-CS. We use $r = 10\%$ edge dropping rate on all three heterogeneous graph datasets.

Number of Base GATs and HetGATs. We use an ensemble of 20 base models for both Ensemble-GAT and Ensemble-HetGAT.

GAT. We use a two-layer model with 64 hidden units in each layer. One attention head is used for both layers[4]. There are two dropouts in GAT: one is the dropout after each propagation layer and the other is the attention dropout. The layer and attention dropout rates are set to (0.8, 0.8) for Cora and Citeseer, (0.8, 0.6) for PubMed, and (0.8, 0.2) for Coauthor-CS. Adam optimizer is used with a learning rate of 0.01, and weight decay of 5e-4 for all four homogeneous datasets. The early stopping patience number is set to 200, and the number of maximum epochs is set to 1000.

APPNP and **DAGNN.** We follow the same settings as in [16] and [13] for APPNP and DAGNN, respectively.

HetGAT. We use a two-layer model with 64 hidden units in each layer. One attention head is used for all three heterogeneous datasets. There are two dropouts: one is the dropout after the feature projection layer, and the other is the dropout after each intermediate layer. Corresponding dropout rates are (0.8, 0.2) for all three heterogeneous datasets. Adam optimizer is used with a learning rate of 0.005, and weight decay of 5e-5 for all three heterogeneous datasets. The early-stopping patience number is set to 100, and the number of maximum epochs is set to 500.

HAN. We follow the same settings as in [18].

SimpleHGN. We use a two-layer model for the ACM and IMDB datasets, and a five-layer model for the DBLP dataset. We use 64 hidden units in each layer, and one attention head. The feature

[4] The original paper [15] uses 8 heads for the first layer, but we find that 1 head already achieves matching results.

URL: http://dx.doi.org/10.14738/tmlai.103.12399

and attention dropout rates are both set to 0.5 for all three heterogeneous datasets. The negative slope is set to 0.05, and β is set to 0.05 for all three heterogeneous datasets. Adam optimizer is used with a learning rate of 0.01 and weight decay of 0 for all three heterogeneous datasets. The early-stopping patience number is set to 100, and the number of maximum epochs is set to 500.

RESULTS

We perform semi-supervised node classification tasks on all benchmark datasets, and take 20 runs of each model on each dataset. The results on the four homogeneous datasets are represented as the average classification accuracy $\pm$ standard deviation, and are summarized in Table 3. The results on the three heterogeneous datasets are represented as the average Macro-F1 score $\pm$ standard deviation, and are summarized in
Table 4.

The proposed ensemble technique significantly improves the performance of GAT and HetGAT. Ensemble-GAT outperforms the state-of-the-art APPNP and DAGNN models on the Cora, Citeseer, and Coauthor-CS datasets as seen in Table 3. The proposed HetGAT already outperforms the state-of-the-art SimpleHGN on the IMDB and DBLP datasets. Ensemble-HetGAT outperforms SimpleHGN and HAN and their ensembles on all three heterogeneous benchmark datasets as seen in
Table 4.

Table 3 Average classification accuracy $\pm$ one standard deviation (in percent) after 20 runs on Cora, Citeseer, PubMed, and Coauthor-CS

Method	Cora	Citeseer	PubMed	Coauthor-CS
GAT	82.9 ± 0.9	71.4 ± 0.9	77.4 ± 0.8	89.9 ± 0.5
Ensemble-GAT	$\mathbf{84.2 \pm 0.5}$	$\mathbf{72.7 \pm 0.5}$	78.1 ± 0.4	$\mathbf{92.2 \pm 0.2}$
APPNP	83.4 ± 0.4	71.8 ± 0.4	79.5 ± 0.2	90.8 ± 0.6
DAGNN	84.0 ± 0.6	72.7 ± 0.7	$\mathbf{79.9 \pm 0.4}$	90.5 ± 0.8

Table 4 Average Macro-F1 score $\pm$ one standard deviation (in percent) after 20 runs on ACM, IMDB, and DBLP

Method	ACM	IMDB	DBLP
HetGAT	91.3 ± 0.8	58.5 ± 1.1	94.0 ± 0.6
Ensemble-HetGAT	$\mathbf{92.4 \pm 0.2}$	$\mathbf{59.5 \pm 0.4}$	$\mathbf{94.3 \pm 0.2}$
HAN	91.2 ± 0.2	57.5 ± 0.6	92.5 ± 0.3
Ensemble-HAN	91.6 ± 0.2	58.7 ± 0.3	92.5 ± 0.1
SimpleHGN	91.4 ± 0.4	56.4 ± 1.5	93.9 ± 0.4
Ensemble-SimpleHGN	91.9 ± 0.2	57.8 ± 0.4	94.1 ± 0.1

To better illustrate the improvements of model performance with the proposed ensemble technique, we plot the results of the ensemble models and their backbone models side by side. We normalize the average accuracy of the Ensemble-GAT by that of GAT considering the different scales of the results on all four homogeneous graphs as we are displaying them in the

same plot. The results on the homogeneous graphs are presented in Figure 2. The ensemble strategy improves the performance of GAT by 1.0% to 2.6% on all four homogeneous benchmark graphs.

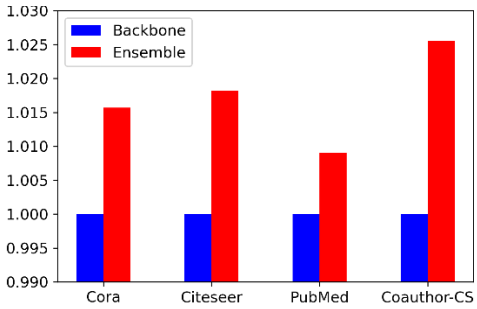


Figure 2 Compare the normalized average classification accuracy of Ensemble-GAT with GAT on Cora, Citeseer, PubMed, and Coauthor-CS

In addition to HetGAT, we also apply the proposed ensemble strategy to HAN and SimpleHGN. We plot the results of the three ensemble HGNNs and their backbone models side by side for all three heterogeneous graphs. The results are presented in Figure 3. The proposed ensemble strategy improves the performance of HetGAT by 0.3% to 1.7% on all three heterogeneous benchmark graphs. The proposed ensemble strategy improves the performance of HAN by 0 to 2.1% on all three heterogeneous benchmark graphs. The proposed ensemble strategy improves the performance of SimpleHGN by 0.2% to 2.5% on all three heterogeneous benchmark graphs.

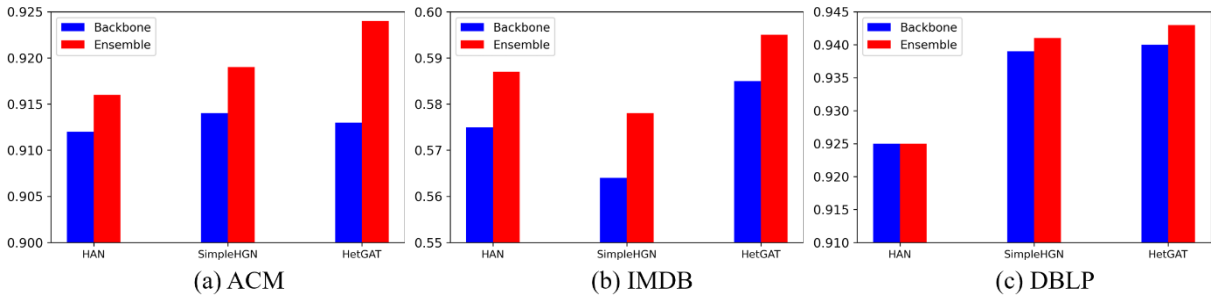


Figure 3 Compare the average Macro-F1 score of Ensemble-HetGAT, Ensemble-HAN, Ensemble-SimpleHGN with their backbone models on (a) ACM, (b) IMDB, (c) DBLP

The ensemble strategy also improves the robustness of GAT and HetGAT, as seen from the reduced variance of the model performance in Table 3 and
Table 4.

Effect of Edge Dropping Rate and Number of Ensembles

In this section, we evaluate the model performance of Ensemble-GAT and Ensemble-HetGAT with difference edge dropping rates r and different number of ensembles.

Edge Dropping Rate

We fix the number of ensembles to 20 base models, and evaluate the ensemble model performance with an edge dropping rate of 10%, 20%, and 50%, respectively. The results averaged by 20 runs of each model are illustrated in Figure 4. We normalize the average accuracy of Ensemble-GAT and Macro-F1 score of Ensemble-HetGAT by dividing them by that

URL: http://dx.doi.org/10.14738/tmlai.103.12399

of the corresponding backbone models for easy comparison considering different scales of the results on different datasets.

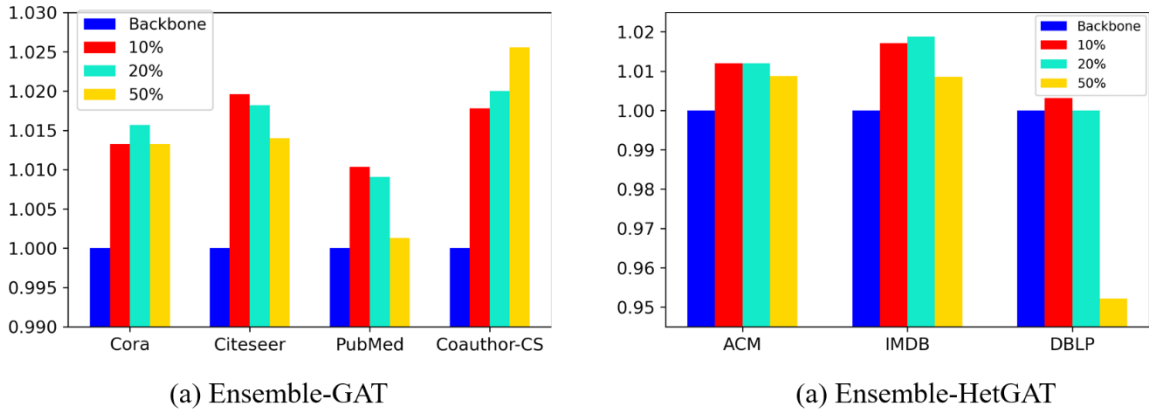


Figure 4 Normalized performance of (a) Ensemble-GAT and (b) Ensemble-HetGAT with different edge dropping rates (10%, 20%, 50%) on benchmark datasets

The ensemble strategy significantly improves GAT on all four homogeneous graph datasets with a 10% and 20% edge dropping rates. 50% dropping rate enhances GAT the most on the Coauthor-CS dataset, while the improvement is limited on the PubMed dataset.

The ensemble strategy significantly improves HetGAT on all three heterogeneous graph datasets with a 10% and 20% edge dropping rates except for the DBLP dataset on which the improvement with a 20% edge dropping rate is limited. The Ensemble-HetGAT with even a 50% dropping rate improves the model performance on the ACM and IMDB datasets, although not as much as with a 10% and 20% dropping rate.

Number of Ensembles

We fix the edge dropping rate to 20% for all datasets, and evaluate the ensemble model performance with 10, 20, and 30 ensembles, respectively. The results averaged by 20 runs of each model are illustrated in Figure 5. We normalize the average accuracy of Ensemble-GAT and Macro-F1 score of Ensemble-HetGAT by dividing them by that of the corresponding backbone models for easy comparison considering different scales of the results on different datasets.

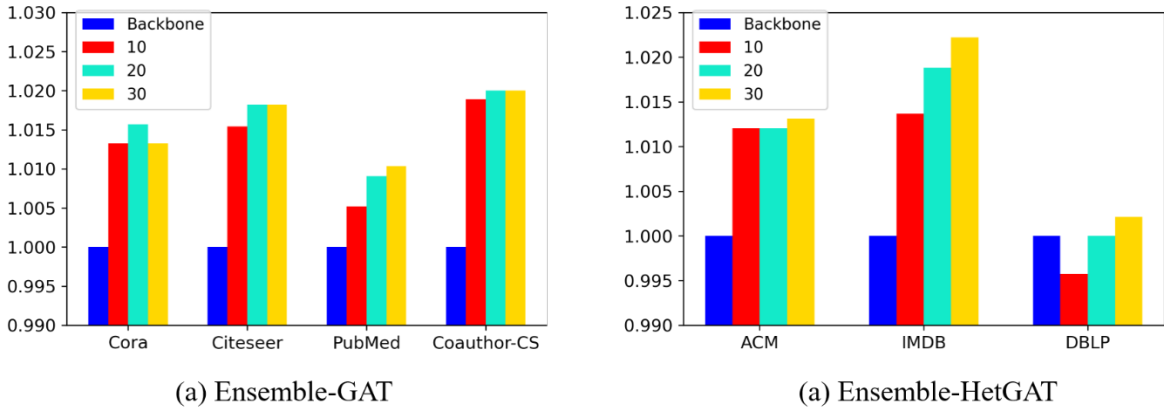


Figure 5 Normalized performance of (a) Ensemble-GAT and (b) Ensemble-HetGAT with different number of ensembles (10, 20, and 30) on benchmark datasets

The improvement of ensemble models is generally increasing with a greater number of ensembles, nevertheless the improvement will be limited when the number of ensembles is large enough depending on the dataset. For the four homogeneous datasets, 20 ensembles of GAT are adequate. The improvement with 30 ensembles of GAT is very limited as compared to 20 ensembles.

Ablation Study

GAT is known as a shallow model whose best performance is usually achieved with a two-layer model [19, 20]. Its performance degrades significantly when stacking multiple propagation layers due to the over-smoothing issue [20]. HetGAT suffers from the similar problem as the core of HetGAT is an edge-specific GAT layer. In this section, we examine whether the ensemble strategy can alleviate the over-smoothing problem in GAT and HetGAT. We evaluate the performance of Ensemble-GAT and Ensemble-HetGAT with a 10% and 20% edge dropping rate at different model depths, and compare the results with the corresponding backbone models. The results are presented in Figure 6 and Figure 7. It is seen that the ensemble strategy slows down the speed of over-smoothing.

CONCLUSION

Existing works on graph neural networks (GNNs) have been dedicated to designing new network architectures and algorithms, but overlooked the application of ensemble techniques to existing models. We propose a simple yet effective bagging-based ensemble strategy, and demonstrate that Ensemble-GAT and Ensemble-HetGAT outperform the state-of-the-art GNN and HGNN models. The proposed ensemble strategy also alleviates the over-smoothing problem in GAT and HetGAT. In addition to GAT and HetGAT, the proposed ensemble strategy can be applied to any backbone GNNs and HGNNs. We demonstrate that the proposed ensemble strategy can enhance the state-of-the-art heterogeneous model – SimpleHGN and the popular heterogeneous model – HAN. Future work could include pursuing the theoretical foundation for the superior performance of Ensemble-GAT and Ensemble-HetGAT, experimenting on other backbone GNNs and HGNNs, and exploring other effective ensemble techniques.

URL: http://dx.doi.org/10.14738/tmlai.103.12399

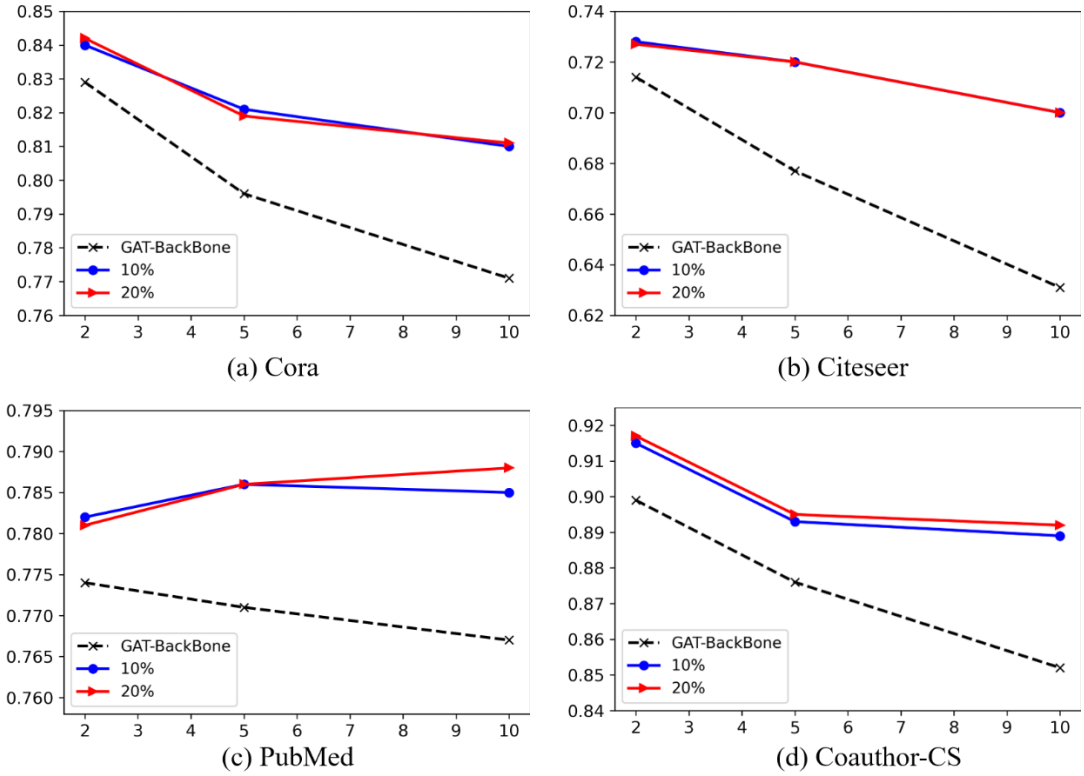


Figure 6 Performance of Ensemble-GAT at different model depths on (a) Cora, (b) Citeseer, (c) PubMed, (d) Coauthor-CS

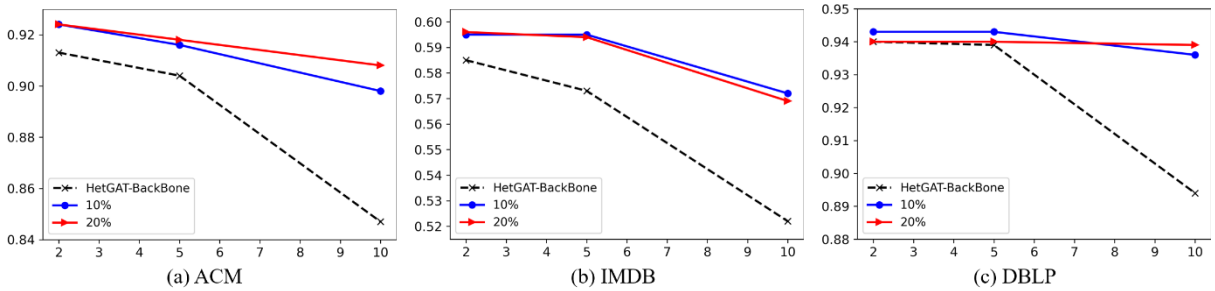


Figure 7: Performance of Ensemble-HetGAT at different model depths on (a) ACM, (b) IMDB, and (c) DBLP

References

Zhang, C. and Y. Ma, *Ensemble machine learning: methods and applications*2012: Springer.

Breiman, L., *Bagging predictors.* Machine learning, 1996. 24(2): p. 123-140.

Freund, Y. and R.E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of computer and system sciences, 1997. 55(1): p. 119-139.

Schapire, R.E., *The strength of weak learnability.* Machine learning, 1990. 5(2): p. 197-227.

Wolpert, D.H., *Stacked generalization.* Neural networks, 1992. 5(2): p. 241-259.

Jacobs, R.A., et al., *Adaptive mixtures of local experts.* Neural computation, 1991. 3(1): p. 79-87.

Jordan, M.I. and R.A. Jacobs, *Hierarchical mixtures of experts and the EM algorithm.* Neural computation, 1994. 6(2): p. 181-214.

Breiman, L., *Random forests.* Machine learning, 2001. 45(1): p. 5-32.

Ho, T.K., *The random subspace method for constructing decision forests.* IEEE transactions on pattern analysis and machine intelligence, 1998. 20(8): p. 832-844.

Friedman, J.H., *Greedy function approximation: a gradient boosting machine.* Annals of statistics, 2001: p. 1189-1232.

Sun, K., Z. Zhu, and Z. Lin, *Adagcn: Adaboosting graph convolutional networks into deep models.* arXiv preprint arXiv:1908.05081, 2019.

Du, J., et al., *Topology adaptive graph convolutional networks.* arXiv preprint arXiv:1710.10370, 2017.

Liu, M., H. Gao, and S. Ji. *Towards deeper graph neural networks*. in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2020.

Wang, M., et al., *Deep graph library: A graph-centric, highly-performant package for graph neural networks.* arXiv preprint arXiv:1909.01315, 2019.

Veličković, P., et al., *Graph attention networks.* arXiv preprint arXiv:1710.10903, 2017.

Klicpera, J., A. Bojchevski, and S. Günnemann, *Predict then propagate: Graph neural networks meet personalized pagerank.* arXiv preprint arXiv:1810.05997, 2018.

Lv, Q., et al. *Are we really making much progress? Revisiting, benchmarking and refining heterogeneous graph neural networks*. in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021.

Wang, X., et al. *Heterogeneous graph attention network*. in *The world wide web conference*. 2019.

Chen, M., et al. *Simple and deep graph convolutional networks*. in *International Conference on Machine Learning*. 2020. PMLR.

Wang, G., et al., *Improving graph attention networks with large margin-based constraints.* arXiv preprint arXiv:1910.11945, 2019.

URL: http://dx.doi.org/10.14738/tmlai.103.12399