You Need to Look Globally: Discovering Representative Topology Structures to Enhance Graph Neural Network

Huaisheng Zhu[1], Xianfeng Tang[2], TianXiang Zhao[1], and Suhang Wang[1(✉)]

[1] Pennsylvania State University, University Park, PA 16802, USA
{hvz5312,tkz5084,szw494}@psu.edu
[2] Amazon, Palto Alto, CA, USA
xianft@amazon.com

Abstract. Graph Neural Networks (GNNs) have shown great ability in modeling graph-structured data. However, most current models aggregate information from the local neighborhoods of a node. They may fail to explicitly encode global structure distribution patterns or efficiently model long-range dependencies in the graphs; while global information is very helpful for learning better representations. In particular, local information propagation would become less useful when low-degree nodes have limited neighborhoods, or unlabeled nodes are far away from labeled nodes, which cannot propagate label information to them. Therefore, we propose a new framework GSM-GNN to adaptively combine local and global information to enhance the performance of GNNs. Concretely, it automatically learns representative global topology structures from the graph and stores them in the memory cells, which can be plugged into all existing GNN models to help propagate global information and augment representation learning of GNNs. In addition, these topology structures are expected to contain both feature and graph structure information, and they can represent important and different characteristics of graphs. We conduct experiments on 7 real-world datasets, and the results demonstrate the effectiveness of the proposed framework for node classification.

Keywords: Graph Neural Network · Global · Node Classification

1 Introduction

Over the past few years, Graph Neural Networks (GNNs) [9,11,19] have shown great success in modeling graph data for a wide range of applications such as social networks [17]. GNNs typically follow the message passing mechanism, which aggregates the neighborhood representation of a node to enrich the node's representation. Hence, the learned representations capture both local topology information and node attributes, which benefits various tasks [11].

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-33377-4_4.

H. Kashima et al. (Eds.): PAKDD 2023, LNAI 13936, pp. 40–52, 2023.
https://doi.org/10.1007/978-3-031-33377-4_4

Despite the success of GNNs in modeling graphs, most of them can only help nodes aggregate local neighbors' information. *First*, long-range or global information can be used to learn better representations. For example, two structurally similar nodes can offer strong predictive power to each other but might be very distant from each other [6]. *Second*, in node classification tasks, we only have partially labeled nodes on graphs. Nodes are often sparsely labeled as it is time-consuming, expensive and sometimes requires domain knowledge to label. In this case, labeled nodes may only propagate their label information to their local neighbors based on local aggregation, which may result in the misclassification of nodes distant from labeled nodes [5]. Therefore, it is important to design GNNs to capture global information and long-range dependencies. Several works [1,12] about aggregating node information from a wider range has been proposed to improve the expressive power of GNNs. However, methods of aggregating information from a wider range cannot explicitly distinguish relevant nodes from lots of distant neighborhoods, which will also result in over-smoothing issues. Thus, how to capture global information needs further investigation.

In real-world graphs, for each class, there are usually some representative ego-graphs. For each ego-graph, it contains one central node and its neighbors from the original graph together with their edge relations, which would be helpful to provide global information about each class in the graph. For example, for malicious account detection, one representative pattern for malicious accounts is that they tend to connect to each other and also try to connect to benign accounts to pretend that they are benign accounts; similarly, benign accounts also have several representative graph patterns. Therefore, it's important to extract and use global representative ego-graphs to improve the performance of GNN models. Though promising, the work on extracting global patterns to facilitate GNN representation learning is rather limited [20]. However, MemGCN [20] only learns global feature information but loses graph structure information.

Therefore, in this paper, we study a novel problem of learning global representative patterns to improve the performance of GNNs. There are several critical challenges: (i) how can we efficiently extract both different structures and features as global information automatically? (ii) how can we make all nodes or even nodes with low-degree to find and utilize highly relevant global information? (iii) how can we use these extracted ego-graphs to improve current GNN models? To fill this gap, we propose a novel framework Graph Structure Memory Enhanced Graph Neural Network, GSM-GNN. It utilizes a clustering algorithm to select representative ego-graphs from the original graph and they are stored in memories. Then, query vectors are generated by preserving topology and node feature information, and they can be used to find relevant global information. Finally, based on query vectors, relevant global information from stored ego-graphs is obtained, and neighborhood patterns about the representative graph structure are also used to augment current GNNs. The main contributions are as follows:

- We study a problem with using global patterns in the graph to improve the performance of GNNs based on local aggregation.
- We develop a novel framework that extends current GNNs with global information. The adoption of memories learns and propagates both global feature and structure information to enrich the representation of nodes.

– Experimental results on seven real-world datasets demonstrate the effectiveness of the proposed approach.

2 Related Works

Graph Neural Networks. Graph Neural Networks have shown great success for various applications such as social networks [11,17]. Generally, existing GNNs can be categorized into two categories, i.e., spectral-based [3,11] and spatial-based [1,9,19]. Spectral-based approaches are defined according to graph signal processing [3]. A first-order approximation is utilized to simplify the graph convolution via GCN [11]. Spatial-based GNN models aggregate information of the neighbor nodes [9]. Despite differences between spectral-based and spatial-based approaches, most GNN variants can be summarized with the message-passing framework [7]. The high-level idea of the message passing mechanism is to propagate the information of nodes through the graph based on pattern extraction and interaction modeling within each layer. However, most of these works only utilize local neighbors' information. Thus, many works about utilizing global information or high-order neighbors for GNNs [1,12] were proposed. There is still little work on using global ego-graph patterns for node classification tasks.

Memory Augmented Neural Networks. Memory Augmented Neural Networks use the memory mechanism with differentiable reading operations to store past experiences and show advantages in many applications [20]. Their implementations of memory on different tasks are inspired by key-value memory [16] and array-structured memory [8]. Also, there have been several works on GNNs that utilized memory-based design for different tasks recently [4,20]. For node classification, memory nodes are introduced in [20] to store global characteristics of nodes, which can learn high-order neighbors' information in the message passing process. In summary, their memory mechanisms try to store important node feature information to improve models' performance. However, there are also important global graph structure patterns like ego-graphs with node feature and their edge relationships. Unlike the aforementioned approaches, our proposed GSM-GNN can learn global ego-graph patterns.

3 Problem Formulation

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ to denote an attributed graph, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes, $\mathcal{E}$ is the set of edges and $\mathbf{X}$ is the attribute matrix for nodes in $\mathcal{G}$. The i-th row of $\mathbf{X}$, i.e., $\mathbf{x}_i \in \mathbb{R}^{1\times d_0}$, is the d_0 dimensional features of node v_i. $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix. $A_{ij} = 1$ if node v_i and node v_j are connected; otherwise $A_{ij} = 0$. We denote a k-hop ego-graph centered at node v_i as $g_i(\mathcal{V}_{g_i}, \mathbf{A}_{g_i})$, where $\mathcal{V}_{g_i}$ include v_i and the set of nodes within k-hop distance with v_i, $\mathbf{A}_{g_i}$ is the corresponding adjacency matrix of the ego-graph.

In real-world graphs, for each class, there are usually some representative ego-graphs, which would represent global information of the graph. Thus, in this paper, we utilize memory mechanisms to learn and store representative ego-graphs and then propagate this information through the whole graph. Memory

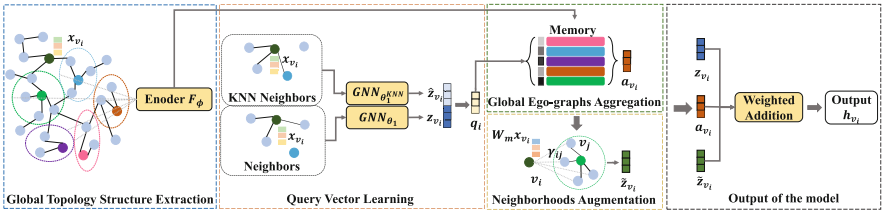

Fig. 1. An illustration of the proposed GSM-GNN.

can be seen as an array of objects, where each object in our paper represents a ego-graph. Each ego-graph $g^i(\mathcal{V}_{m_i}, \mathbf{A}_{m_i})$ is centered at node v'_{m_i} with its k-hop neighbors drawn from the original graph $\mathcal{G}$, where $\mathcal{V}_{m_i} = \left\{v^1_{m_i}, \ldots, v^{B'_i}_{m_i}\right\}$ and each node in this set is from $\mathcal{V}$ with their edges in the adjacency matrix $\mathbf{A}_{m_i}$. B'_i is the number of nodes in memory i and $\mathbf{A}_{m_i} \in [0,1]^{B'_i \times B'_i}$ to represent the adjacency matrix for ego-graph g_i. Note that memories are updated during the training process. Our memory module in the last epoch is defined as $\mathcal{M}_T = \{g^1_T(\mathcal{V}_{m_1}, \mathbf{A}_{m_1}), \ldots, g^B_T(\mathcal{V}_{m_B}, \mathbf{A}_{m_B})\}$, where B is the number of memories, T is the number of training epochs. In semi-supervised node classification, only a subset of nodes are labeled. We denote the labeled set as $\mathcal{V}_L \in \mathcal{V}$ with $\mathcal{Y}_L$ being the corresponding label set of the labeled nodes. The remaining nodes $\mathcal{V}_U = \mathcal{V} \backslash \mathcal{V}_L$ are the unlabeled set. The problem is formally defined as:

Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \mathbf{X})$ *and the partial labels* $\mathcal{Y}_L$*, we aim to learn a node classifier* Q_θ *via plugging our proposed memories into current GNNs.* Q_θ *should model edges together with node features accordingly via learning global ego-graphs during the label prediction process* $Q_\theta(\mathcal{V}, \mathbf{A}, \mathbf{X}) \rightarrow \mathcal{Y}$.

4 Methodology

In this section, we introduce the details of the proposed GSM-GNN, which stores representative ego-graph structures and propagates their information to enhance the representation learning of GNNs. An overview is shown in Fig. 1. Our model can be split into two parts: Global Topology Structure Extraction, and Graph Structure Memory Augmented Representation Learning. Firstly, we store global topology structure information in memories by nodes with their neighborhoods as ego-graphs from the center of clusters, which are obtained via the clustering algorithm on both original and learnt nodes features. Then, we use the query vectors based on feature and structure information to obtain relevant global information from memories, which can enhance the expressive power of GNNs based on local aggregation. The details of them will be introduced below.

4.1 Global Topology Structure Extraction

To mitigate limits of GNNs based on local aggregation, we propose to extract global information from the original graph to enhance GNN models. For graph

data, it doesn't only contain node features but also edge relations between nodes. Thus, our model is to extract representative ego-graphs, which preserve both global topology structure together with node feature information, and store them into memories. Then, these ego-graphs are used to enhance the representation learning of GNNs which is introduced in the next section. To extract these ego-graphs, we do k-Medoids clustering on nodes based on their original features $\mathbf{X}$. We then select B central nodes of clusters. Firstly, nodes in different clusters have different information about the graph so a set of clusters are related to different global patterns, and central nodes represent important characteristics of clusters. Furthermore, information in one single node is limited, and neighbors of central nodes and their relationship patterns are also important. Thus, these central nodes with their k-hop neighbors, which form ego-graphs, are treated as global topology graph structure information and stored in a memory set $\mathcal{M}_0$.

However, knowledge of extracting representative ego-graphs only on the original feature is limited so it is necessary to update stored ego-graph based on more informative representation vectors during the training process. Therefore, we use k-Medoids clustering algorithms on the hidden representation $\mathbf{Z}$ which is the output representation of GNN models. Also, central nodes may represent the characteristics of some global patterns in the whole graph. Central nodes and their one-hop or two-hop neighbors as ego-graphs are stored in $\mathcal{M}_t$ at the training epoch t. In our experiment, memories are updated every 100 epochs. An update of memories is utilized to automatically extract representative ego-graph patterns on the whole graph. Then, ego-graphs stored in the memory are used to enhance the representation learning of GNNs.

4.2 Graph Structure Memory Augmented Representation Learning

By extracting and storing representative ego-graphs, we propose to propagate their global information to enhance representation learning of GNNs. The high level idea is to use stored ego-graphs in the memory to improve the expressive power of GNNs. To achieve this purpose, we need to query and aggregate relevant global information from memories and use it to augment representation learning of GNNs. And it can be split into three parts: (1) **Query Vector Learning**, which learns the query vector of each node, and it will be used to obtain relevant global information; (2) **Global Ego-graphs Aggregation**, which encodes ego-graphs and aggregate their information by the similarity between query vectors and encoded feature vectors of ego-graphs in memories; and (3) **Neighborhoods Augmentation**, which utilizes the neighborhood patterns to augment the representation, and can add long-range interactions for distant nodes through this way. Specifically, we firstly obtain representation vectors $\mathbf{z}_{v_i}$ for node v_i via one-layer GNN with local aggregation by:

$$\mathbf{z}_{v_i} = GNN_{\theta_1}(\mathbf{X}, \mathbf{A})_{v_i}, \tag{1}$$

where GNN can be flexible to different GNNs like GCN, GraphSage and GAT.

Query Vector Learning. The purpose of our model is to use ego-graphs stored in memories to enhance the representation learning of GNNs. But for a node v_i, not all memory elements are relevant to v_i. Thus, we first need to learn query vectors of nodes to get relevant information in memories. To guarantee that nodes can query relevant memories, query vector learning should preserve nodes' features and structural information. Firstly, local neighborhood patterns are represented as a vector z_{v_i} with the message passing process of GNNs and can be immediately treated as the query vector for node v_i. However, nodes in the graph only have a small number of neighbors and their local neighborhood patterns may contain bias. Thus, we further augment the query vector with feature information, which can help low-degree nodes be more representative [10]. Concretely, a KNN graph is constructed based on the cosine similarity. For a given node pair (v_i, v_j), their cosine similarity is calculated as $\mathbf{O}_{ij} = \frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\cdot\|_2 \|\mathbf{x}_j\|_2}$. We choose $k \in \{20, 30\}$ nearest neighbors via cosine similarity for each node and get a KNN graph. The adjacency matrix of the KNN graph is denoted as $\hat{A}$. Then, node similarity information is aggregated via GNN on KNN graph as:

$$\hat{\mathbf{z}}_{v_i} = GNN_{\theta_1^{KNN}}(\mathbf{X}, \hat{\mathbf{A}})_{v_i}, \tag{2}$$

where θ_1^{KNN} is the learnable parameter for learning KNN graph information, which is denoted as $\hat{\mathbf{z}}_{v_i}$. To learn query vectors from feature similarity and structural information from $\mathbf{z}_{v_i}$ and $\hat{\mathbf{z}}_{v_i}$, these two vectors are concatenated together and used as an input for an MLP layer to obtain the query vector as:

$$\mathbf{q}_i = [\mathbf{z}_{v_i} \| \hat{\mathbf{z}}_{v_i}]\mathbf{W}_q + \mathbf{b}_q \tag{3}$$

where $\mathbf{W}_q \in R^{2d\times d}$ and $\mathbf{b}_q \in R^d$ are learnable parameters, d is the dimension of vectors $\mathbf{z}_{v_i}, \hat{\mathbf{z}}_{v_i}$. Then, $\mathbf{q}_i$ will be used to query relevant global information.

Global Ego-Graphs Aggregation. Furthermore, to query relevant ego-graphs in memories via $\mathbf{q}_i$, we also need to encode the ego-graph information into vectors. To achieve this goal, we utilize one-layer GCN to obtain representation vectors for all ego-graphs in memories:

$$\mathbf{m}_i = F_\phi(\mathbf{X}_{m_i}, \mathbf{A}_{m_i}), \tag{4}$$

where $F_\phi(*)$ represents one-layer GCN with graph pooling to get the representation vectors of ego-graphs in memories, $\mathbf{m}_i \in \mathbb{R}^{1\times d}$. $\mathbf{X}_{m_i}$ is the representation matrix for nodes in memory i, where $\mathbf{X}_{m_i}[j,:] \in \mathbb{R}^{1\times d_0}$ is the representation vectors of node $v_{m_i}^j$ which is obtained from the original feature matrix $\mathbf{X}$. Note that the pooling method here is the mean pooling method. Then we calculate the similarity between $\mathbf{q}_i$ and $\mathcal{M}_t$ in the layer l of the training epoch t as:

$$\mathbf{s}_i = Softmax(\mathbf{q}_i(\mathbf{M})^T), \tag{5}$$

where $\mathbf{M} \in \mathbb{R}^{B_i'\times d}$ is the representation matrix for B_i' memories. The similarity scores measure the importance of each global pattern in the memory. Any pattern with a higher attention score is more similar to the local structural patterns

of nodes. The representation vector of global information for node v_i is then constructed from the weighted sum of all global patterns in the memory $\mathcal{M}_t$ as

$$\mathbf{a}_{v_i} = \sum\nolimits_{j=1}^{B} s_{i,j}\mathbf{m}_j, \tag{6}$$

where $\mathbf{a}_{v_i}$ represents global ego-graph information of node v_i. $s_{i,j}$ is the similarity score between node i and memory j.

Neighborhoods Augmentation. Each memory module contains one representative central node with its one or two-hop neighbors and their edge relations. Even though the central node of each memory node has aggregated information from its neighbors, we will also lose some information about their neighbors. Thus, it's important to explore neighborhood distributions from relevant ego-graphs, which can further capture ego-graphs' information. However, it will be time-consuming and introduce more noisy neighbors' information if we add all neighbors' of nodes in memories to one node's augmented neighbors. Thus, we select the most relevant memory modules as $r_i = \arg\max_j s_{i,j}$, where r_i is the index of the most relevant memory module. Then, we obtain the most relevant ego-graph $g_t^{r_i}(\mathcal{V}_{m_{r_i}}, \mathbf{A}_{m_{r_i}})$ of the training epoch t based on the similarity of structural information between local ego-graph patterns and global ego-graph patterns. Central nodes have aggregated their neighbors' information and are treated as representation vectors in memories which are added as global information in Eq. (6). Instead of aggregating central nodes' information again, neighborhood nodes of them in $\mathcal{V}_{m_{r_i}}$ are treated as the augmented neighbors for the node i and their information is aggregated for node v_i. Enhanced neighbors may contain noisy information so an attention mechanism is utilized to assign different weights to augmented neighbors:

$$\gamma_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{u}^T\left[\mathbf{x}_{v_i}\mathbf{W}_m \| \mathbf{x}_{v_j}\mathbf{W}_m\right]\right)\right)}{\sum_{k\in\mathcal{V}_{m_{r_i}}} \exp\left(\text{LeakyReLU}\left(\mathbf{u}^T\left[\mathbf{x}_{v_i}\mathbf{W}_m \| \mathbf{x}_{v_k}\mathbf{W}_m\right]\right)\right)}, \tag{7}$$

where $\mathbf{W}_m \in \mathbb{R}^{d_0\times d}$ is the learnable parameters, $\mathbf{u} \in \mathbb{R}^{1\times 2d}$ is the learnable weight vector, γ_{ij} represents the weight for nodes j in $\mathcal{V}_{m_{r_i}}$ when node i aggregates information from node j. To mitigate noise from augmented neighbors, we aggregate these neighbors via the weight γ_{ij}. The aggregation process of augmented neighbors is:

$$\tilde{\mathbf{z}}_{v_i} = \sum_{j\in\mathcal{V}_{m_{r_i}}\setminus v'_{m_{r_i}}} \gamma_{ij}\mathbf{x}_{v_j}\mathbf{W}_m, \tag{8}$$

where $v'_{m_{r_i}}$ is the central node of the ego-graph in the memory r_i.

Finally, we get the local representation $\mathbf{z}_{v_i}$, global representation $\mathbf{a}_{v_i}$ and augmented neighbors representation $\tilde{\mathbf{z}}_{v_i}$. Different nodes might rely on different information. For example, low degree nodes may need more global information; while high degree nodes with enough similar neighborhoods information may

only need more local information. Thus, we propose to assign different weights to get the final node representation. Specifically, the weight is calculated as:

$$\boldsymbol{\beta}_i = Softmax(\mathbf{W}_o[\tilde{\mathbf{z}}_{v_i}||\mathbf{a}_{v_i}||\mathbf{z}_{v_i}] + \mathbf{b}_o), \tag{9}$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ are learnable parameters, $\boldsymbol{\beta}_i \in \mathbb{R}^3$ is the weight for different representation vectors. Then, different representation vectors are added together based on their weights to get $\mathbf{h}_{v_i}$ as:

$$\mathbf{h}_{v_i} = \beta_{i,0}\mathbf{z}_{v_i} + \beta_{i,1}\mathbf{a}_{v_i} + \beta_{i,2}\tilde{\mathbf{z}}_{v_i}. \tag{10}$$

With the above operation, GNN models can help nodes aggregate both local neighborhood information and global information adaptively from memories. Therefore, our memory modules can store representative global information and help propagate this information on the whole graph. Note that our memory module can be also added in more layers, to reduce the computational cost, we only use it to augment the representation of one layer.

4.3 Objective Function of GSM-GNN

With the representation $\mathbf{H}$ capturing both local and global information, we add another GNN layer together with Softmax function to predict the class probability vector of each node v_i as:

$$\hat{\mathbf{y}}_{v_i} = Softmax(GNN_{\theta_p}(\mathbf{H}, \mathbf{A})_{v_i}) \tag{11}$$

where $\hat{\mathbf{y}}_{v_i}$ is the predicted label's probability by passing the output from the final GNN layer to a softmax function. θ_p represents the parameters of our model's final prediction layer. The cross-entropy loss function for node classification is:

$$\min_{\theta} \mathcal{L}_c(\theta) = -\sum_{v_i \in \mathcal{V}_L} \sum_{c=1}^{C} \mathbf{y}_{v_i}^c \log \hat{\mathbf{y}}_{v_i}^c, \tag{12}$$

where C is the number of classes, $\mathbf{y}_{v_i}$ is the one hot encoding of v_i's label and $\mathbf{y}_{v_i}^c$ is the c-th element of $\mathbf{y}_{v_i}$. θ denotes the set of parameters.

5 Experiments

In this section, we conduct experiments on real-world datasets to demonstrate the effectiveness of GSM-GNN. In particular, we aim to answer the following research questions: (**RQ1**) Can the proposed memory mechanism improve node classification performance? (**RQ2**) Is the designed approach flexible to be applied in various GNN variants? (**RQ3**) What are the contributions of the proposed module in this paper for GSM-GNN?

Table 1. Node classification performance (Accuracy (%) ± Std.) on all graphs.

Method	Cora	Citeseer	Computer	Photo	Physics	Chameleon	Squirrel
MLP	45.44 ±1.55	52.61 ±0.80	67.35 ±0.71	79.10 ±0.74	92.11 ±0.11	48.00 ±1.5	34.02 ±2.13
GCN	74.65 ±1.91	65.20 ±0.74	80.80 ±0.29	87.90 ±0.58	94.24 ±0.10	63.50 ±1.93	47.48 ±2.00
GraphSage	75.43 ±2.08	65.63 ±0.35	73.47 ±0.34	86.31 ±0.15	94.56 ±0.08	48.36 ±2.08	35.88 ±1.20
GAT	71.30 ±0.92	64.55 ±2.32	76.47 ±1.49	85.74 ±1.32	94.21 ±0.07	64.12 ±1.82	48.18 ±4.14
Mixhop	70.40 ±2.83	62.44 ±1.14	75.88 ±1.00	87.92 ±0.53	94.41 ±0.23	60.71 ±1.55	44.11 ±1.10
ADA-UGNN	74.29 ±1.95	65.30 ±1.28	79.88 ±0.92	88.08 ±0.78	94.70 ±0.11	52.19 ±1.85	34.84 ±1.36
H2GCN	72.45 ±0.46	66.10 ±0.48	78.22 ±0.75	86.94 ±0.47	94.59 ±0.09	59.13 ±2.00	36.91 ±1.10
FAGCN	69.53 ±0.12	61.07 ±0.32	81.09 ±0.14	85.33 ±0.12	94.67 ±0.07	65.57 ±4.80	48.73 ±2.50
Simp-GCN	74.24 ±1.32	66.24 ±1.05	74.23 ±0.12	82.41 ±1.36	94.43 ±0.07	64.71 ±2.30	42.81 ±1.20
GCN-MMP	72.23 ±2.19	64.95 ±1.69	79.55 ±1.48	87.56 ±1.03	94.42 ±0.12	66.17 ±1.68	50.93 ±1.45
GSM-GCN	**75.93 ± 0.65**	**66.33 ± 0.49**	**81.32 ± 0.34**	**88.87 ± 0.35**	**94.72 ± 0.07**	**67.20 ± 1.85**	**54.14 ± 1.60**

5.1 Datasets

We conduct experiments on seven publicly available benchmark datasets. **Cora and Citeseer** [11] are two datasets for citation networks **Computers and Photo** [18] are two datasets for the Amazon co-purchase graph [14]. **Physics** [18] is a larger co-authorship graph. **Chameleon and Squirrel** [15] are two datasets for the web pages in Wikipedia. They are used for heterophilous graphs. For Chameleon and Squirrel, we follow the 10 standard splits from [21]. For other datasets, we randomly split the dataset into train/val/test as 2.5%/2.5%/95%. The random split is conducted 5 times and average performance will be reported.

5.2 Experimental Setup

Baselines. We compare GSM-GNN with representative methods for node classification, which includes MLP, GCN [11], GraphSage [9], GAT [19], Mixhop [1], ADA-UGNN [13]. We also compare GSM-GNN with the following representative and state-of-the-arts GNN models on heterophilous graphs, which includes H2GCN [21], FAGCN [2], Simp-GCN [10] and GCN-MMP [4].

Configurations. All experiments are conducted on a 64-bit machine with Nvidia GPU (Tesla V100, 1246 MHz, 16 GB memory). For a fair comparison, we utilize a two-layer neural network for all methods, and the hidden dimension is set as 64. The learning rate is initialized to 0.001. Besides, all models are trained until converging, with the maximum training epoch being 1000. The implementations of all baselines are based on Pytorch Geometric or their original code. For our method, the update epoch of memories is fixed at 100 and B is set by grid search from 5 to 30 for all datasets. The hyperparameters of all methods are tuned on the validation set. We adopt accuracy (ACC) as the metric.

5.3 Performance on Node Classification

In this subsection, we compare the performance of the proposed method with baselines for node classification on the heterophilous and homophilous graphs introduced in Sect. 5.1, which aims to answer **RQ1**. For Cora, Citeseer, Computers, Photo and Physics, each experiment is conducted 5 times and for Chameleon

Table 2. Node classification accuracy with different GNNs.

	Cora	Citeseer	Chameleon	Squirrel
GCN	74.65 ± 1.91	65.20 ± 0.74	64.80 ± 1.93	47.48 ± 0.20
GSM-GCN	75.93 ± 0.65	66.33 ± 0.49	67.20 ± 1.85	54.14 ± 1.60
GraphSage	75.43 ± 2.08	65.63 ± 0.35	48.36 ± 2.08	35.88 ± 1.20
GSM-Sage	76.77 ± 0.62	67.01 ± 0.29	50.59 ± 2.81	37.86 ± 1.47
GAT	71.30 ± 0.92	64.55 ± 2.32	64.12 ± 1.82	48.18 ± 4.14
GSM-GAT	73.38 ± 1.82	65.55 ± 1.20	64.67 ± 1.62	53.51 ± 2.78

and Squirrel, each experiment is conducted 10 times. The average results with standard deviation are reported in Table 1. Note that GSM-GCN uses the GCN as the backbone of our proposed memory modules. From the table, we make the following observations: (1) Compared with GCN and other GNN models, GSM-GCN can consistently improve the performance of GCN on all datasets, which demonstrates the effectiveness of the proposed memory module. Furthermore, GSM-GCN outperforms all baselines on all datasets, especially for the Squirrel dataset. This is because the proposed memory module can capture representative ego-graphs to facilitate GNNs to capture global information. (2) Both Simp-GNN and GSM-GCN utilize the information of node features' similarity. GSM-GCN significantly outperforms Simp-GNN on all datasets. This is because GSM-GCN adopts similarity information as a query vector to query global information, which shows the effectiveness of our query mechanism based on feature similarity. (3) GCN-MMP also designs a memory mechanism on nodes to improve GNNs' performance. The proposed GSM-GCN consistently outperforms GCN-MMP on all datasets. This is because GCN-MMP only stores feature vectors in memories while our method utilizes both feature and structure information.

5.4 Flexibility of GSM-GNN for Various GNNs

To answer **RQ2**, we conduct experiments with different architectures of GSM-GNN by inserting our memory module into different GNN variants. Specifically, we test our memory modules on GCN, GraphSage, and GAT layers. For GCN, GraphSage, and GAT, we utilize a two-layer graph network with 64 hidden dimensions. For a fair comparison, all models use the same settings. For all the methods, hyperparameters are tuned via the performance of the validation set. Each experiment is conducted 5 times on Cora and Citeseer datasets, and 10 times on Chameleon and Squirrel. The average performance with standard deviation is reported in Table 2. From the table, we have the following observations: (i) GSM-GNN can consistently improve the performance on these four datasets with all backbones which indicates that our proposed memory mechanism is effective when incorporated into other GNN variants and demonstrates the flexibility and advantage of the proposed method; and (ii) in particular, the proposed memory module can significantly improve the GCN, GraphSage and

Table 3. Compared with different information from memory modules.

Method	Cora	Citeseer	Computer	Photo	Physics	Chameleon	Squirrel
GCN	74.65 ± 1.91	65.20 ± 0.74	80.80 ± 0.29	87.90 ± 0.58	94.24 ± 0.10	63.50 ± 1.93	47.48 ± 2.00
Without $\mathbf{a}_{v_i}$	75.27 ± 1.26	65.82 ± 0.70	80.93 ± 0.21	86.67 ± 0.29	94.63 ± 0.07	67.09 ± 1.63	54.04 ± 1.20
Without $\tilde{\mathbf{z}}_{v_i}$	73.63 ± 1.59	66.01 ± 0.49	80.87 ± 0.17	88.04 ± 0.34	94.47± 0.40	64.56 ± 2.16	53.87 ± 0.87
GSM-GCN	**75.93 ± 0.65**	**66.33 ± 0.49**	**81.32 ± 0.34**	**88.87± 0.35**	**94.72 ± 0.07**	**67.20 ± 1.85**	**54.14 ± 1.60**

GAT on the heterophilous graphs. This shows that our memory module can use global information to improve current GNNs.

5.5 Ablation Study

To answer **RQ3**, in this section, we conduct an ablation study to evaluate the influence of each queried information from memories including $\mathbf{a}_{v_i}$ and $\hat{\mathbf{z}}_{v_i}$ in GSM-GNN. First, to investigate how the global ego-graph information ($\mathbf{a}_{v_i}$) influences the performance of node classification, we only encode ego-graph information and add it with local information via the attention mechanism. Then, we also query augmented neighbors and aggregate this information as global information $\hat{\mathbf{z}}_{v_i}$. We use GCN as the backbone for ablation studies. The experiments are conducted on all graphs. The average performance in terms of Accuracy is shown in Table 3. From the table, we observe that: (**i**) Comparing GCN with "Without $\mathbf{a}_{v_i}$", augmented neighbors from ego-graphs in memories can lead to a little improvement on almost all graphs. It means that augmented neighbors from memories can provide more similar nodes information during the aggregation process; (**ii**) "Without $\tilde{\mathbf{z}}_{v_i}$" only utilizes ego-graph information in memories and it also performs better than the original GCN. It is because global patterns based on ego-graph may contain important label information which can be used to improve the performance of the original GCN; (iii) Finally, our proposed GSM-GCN has the best performance on all datasets because GSM-GCN can aggregate local information, global information from ego-graphs, and augmented neighbors adaptively. This ablation study further proves the effectiveness of our proposed method to capture global information from the whole graph.

6 Conclusion

In this paper, we study a novel problem of learning global patterns and propagating global information to improve the performance of GNNs. We propose a novel framework Graph Structure Memory Enhanced Graph Neural Network (GSM-GNN) which stores representative global patterns with nodes and graph structure, and can be used to augment the representation learning of GNNs. Through extensive experiments, we validate the advantage of the proposed GSM-GNN, which can utilize a memory network to store and propagate global information.

Acknowledgements. This material is based upon work supported by, or in part by, the National Science Foundation (NSF) under grant number IIS-1909702, the Army Research Office (ONR) under grant number W911NF21-1-0198, and Department of Homeland Security (DNS) CINA under grant number E205949D. The findings in this paper do not necessarily reflect the view of the funding agencies.

References

1. Abu-El-Haija, S., et al.: Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing. In: International Conference on Machine Learning, pp. 21–29. PMLR (2019)
2. Bo, D., Wang, X., Shi, C., Shen, H.: Beyond low-frequency information in graph convolutional networks. arXiv preprint arXiv:2101.00797 (2021)
3. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
4. Chen, J., Liu, W., Pu, J.: Memory-based message passing: Decoupling the message for propogation from discrimination. arXiv preprint arXiv:2202.00423 (2022)
5. Dai, E., Jin, W., Liu, H., Wang, S.: Towards robust graph neural networks for noisy graphs with sparse labels. arXiv preprint arXiv:2201.00232 (2022)
6. Donnat, C., Zitnik, M., Hallac, D., Leskovec, J.: Learning structural node embeddings via diffusion wavelets. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1320–1329 (2018)
7. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR (2017)
8. Graves, A., Wayne, G., Danihelka, I.: Neural turing machines. arXiv preprint arXiv:1410.5401 (2014)
9. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
10. Jin, W., Derr, T., Wang, Y., Ma, Y., Liu, Z., Tang, J.: Node similarity preserving graph convolutional networks. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 148–156 (2021)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
12. Liu, M., Wang, Z., Ji, S.: Non-local graph neural networks. IEEE Trans. Pattern Anal. Mach. Intell. (2021)
13. Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., Shah, N.: A unified view on graph neural networks as graph signal denoising. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management (2021)
14. McAuley, J., Targett, C., Shi, Q., Van Den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (2015)
15. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: GEOM-GCN: geometric graph convolutional networks. arXiv preprint arXiv:2002.05287 (2020)
16. Pritzel, A., et al.: Neural episodic control. In: International Conference on Machine Learning, pp. 2827–2836. PMLR (2017)
17. Qu, L., Zhu, H., Zheng, R., Shi, Y., Yin, H.: ImGAGN: imbalanced network embedding via generative adversarial graph networks. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2021)

18. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
19. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
20. Xiong, T., Zhu, L., Wu, R., Qi, Y.: Memory augmented design of graph neural networks (2020)
21. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Beyond homophily in graph neural networks: Current limitations and effective designs. Adv. Neural. Inf. Process. Syst. **33**, 7793–7804 (2020)