Ego-Aware Graph Neural Network

Zhihao Dong, *Student Member, IEEE*, Yuanzhu Chen, *Member, IEEE*, Terrence S. Tricco, Cheng Li, *Senior Member, IEEE*, and Ting Hu

***Abstract*—In scientific exploration and daily life, interconnectivity between entities is ubiquitous, such as species via the food webs, people among social networks, and products and customers in e-commerce. Graphs and networks are natural constructs to model such linkages, which can capture the relational information in addition to features provided by individual nodes. Graph neural networks (GNNs) are a powerful computation tool for reasoning about interconnected data and can help us make better use of the information imbued in data collectively. GNNs have found many successful applications, such as predicting drug side-effects, classifying diseases, and predicting the function of proteins. Most existing GNN methods are designed for datasets with moderate node features, where individual node features provide limited information, and assortative graph datasets, where nodes and their neighbours are more likely to be similar. However, for graphs with rich node features, or for the disassortative graphs, where nodes and their neighbours tend to be different, the message passing process in GNNs might be susceptible to inferences from the neighbourhood, leading to performance deterioration. To address this issue, we find that ego networks contain an extra layer of information to further distinguish different ego nodes. Based on that, we propose a generic GNN model that can better utilize structural information of nodes' proximity to extract informative messages and resist contradictory ones from their neighbourhood. We analyze how node features and graph structure can influence the performance of GNN models. Experimental results provide insight on how our method outperforms the baselines. The presented model paves the way for incorporating ego networks' structural information into the learned graph representations, which brings GNNs with better performance and higher robustness over different datasets.**

***Index Terms*—Graph neural network, ego aggregation, Jaccard similarity, graph representation learning.**

I. Introduction

THE network, or graph in mathematics, is a powerful tool to represent and analyse complex systems in the real world. Given a set of entities, their individual features, and how they are linked, we often wish to classify a node by examining the graph [1], [2], [3], predict if there may be a missing link between two individuals [4], [5], [6], [7], [8], or even decide whether the graph should possess certain traits overall [9], [10], [11], [12], [13]. The past few decades has seen an evolution of methodology for reasoning about graphs to distill knowledge from networked data. Learning from graphs started with data mining of graph properties available from different scopes of the input graph. The properties include various node degrees and pagerank, motifs and orbits, betweenness centrality, and more [14], [15]. However, traditional learning approaches are limited since they require hand-engineered statistics and measures of the graph, which is inflexible and time-consuming [16]. In recent years, graph representation learning has provided an alternative to reasoning about graphs [17], [18], [19], [20]. Instead of extracting features from the graph manually, graph representation learning automatically maps a graph to vector representations in a Euclidean space encompassing both structural and feature information. Graph neural networks (GNNs), as a typical graph representation learning approach, have attained potent performance in reasoning about networks and attracted significant attention recently [1], [2], [3]. Compared to traditional deep learning paradigms designed for the Euclidean domain, such as convolutional neural networks (CNNs) [21], [22], recurrent neural networks (RNNs) [23], and autoencoders [24], GNNs distill informative vectorial representations from non-Euclidean graphical data. GNNs have found important applications in various fields [25], such as road traffic prediction [26], [27], [28], urban planning [29], drug discovery [30], molecular fingerprint calculation [31], [32], [33], [34], cancer gene prediction [35], and financial fraud detection [36], [37]. Moreover, GNNs can be combined with reinforcement learning and other machine learning techniques to solve combinatorial optimization problems in graphs [38], such as key player identification [39], influence maximization [40], and routing problems [41], [42].

While GNNs have been demonstrated to yield significant performance improvement in many applications, such performance can be impacted by the graph structures to a large degree. When sufficiently utilizing the graph structural information, general GNN models, such as GCN [1], GAT [3], and GraphSAGE [2], can risk over-emphasizing the network's structure and detrimentally affect performance. This can happen either 1) when neighbour interference becomes serious on graphs with rich node features, or 2) when the graph is disassortative. A graph is assortative when links in the graph are mostly likely joining similar nodes. It is called disassortative otherwise. Information aggregation at a node draws heavily from its neighbours

Manuscript received 16 June 2023; revised 12 September 2023; accepted 28 October 2023. Date of publication 6 November 2023; date of current version 23 February 2024. This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Discovery Grants RGPIN-2017-05201 and RGPIN-2018-03792, in part by the IDEaS under Grant MN3-0022, and in part by the Simon Fraser University under Grant 13-2130-90140-N001210. Recommended for acceptance by Dr. Chi Harold Liu. *(Corresponding author: Yuanzhu Chen.)*

Zhihao Dong, Yuanzhu Chen, and Ting Hu are with the School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: z.dong@queensu.ca; yuanzhu.chen@queensu.ca; ting.hu@queensu.ca).

Terrence S. Tricco is with the Department of Computer Science, Memorial University, St. John's, NL A1B 3X5, Canada (e-mail: tstricco@mun.ca).

Cheng Li is with the School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: cheng_li_5_@sfu.ca).

Digital Object Identifier 10.1109/TNSE.2023.3330450

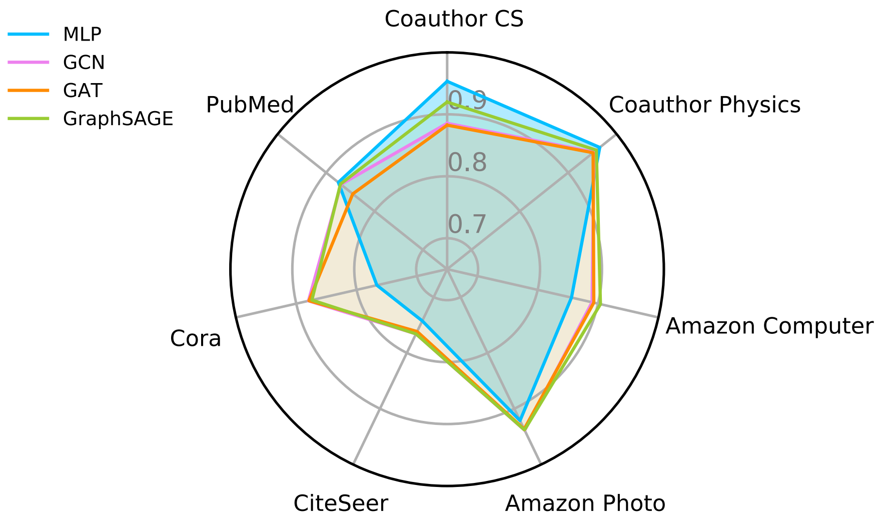


Fig. 1. Accuracy comparison of different models on different datasets.

assuming they are similar to itself and can be of referential values. While such an assumption about graph assortativity may be valid for the majority of networks, disassortative networks cannot be disregarded. When the graph is not sufficiently assortative [43] or nodes are feature-rich, such undiscriminating message passing in GNN can result in destructive aggregation of neighbour information rather than constructive. Naturally, when augmenting node features with graph structural information, one would expect better performance in a reasoning task. However, net performance loss may be possible in the scenarios discussed above.

Fig. 1 illustrates the loss of performance when the existing GNNs are presented with feature-rich datasets. An experiment was conducted that compares the accuracy of node classification among Multilayer perceptron (MLP), GCN, GAT, and GraphSAGE on seven datasets. MLP depends on pure node features, while GCN, GAT, and GraphSAGE utilize both node features and network structure. The detailed setup about the experiment is provided in the Section of Methods. It can be seen that GCN, GAT, and GraphSAGE do not always perform better than MLP (e.g., on the datasets of PubMed, Coauthor CS and Coauthor Physics), even though they use information from neighbours in the network in addition to features of individual nodes. That is, MLP is not enclosed by the graph models in the radar plot, even though more information is available to GCN, GAT, and GraphSAGE.

GNNs generally adopt set aggregation functions, such as sum aggregator and mean aggregator, when collecting information from neighbouring nodes and maintaining order-invariance among them. Meanwhile, these aggregators do not fully leverage the structural information in nodes' neighbourhoods, and the local structure could facilitate more than a simple aggregation. Some approaches, such as GEOM-GCN [43] and Non-local GNN [44], have been designed for disassortative graphs and achieve better performance on those graphs, but they struggle to get a significant improvement on graphs with rich node features, where neighbour interference is serious. Moreover, many graphs are intermediate and do not have an explicit assortativity or disassortativity, so it is difficult to determine whether regular GNNs or disassortative-specific methods should be applied.

In this work, a generic and robust GNN model is proposed that has no requirements for graph assortativity or node feature richness. This GNN model is called *Ego-Aware Graph Neural Network* (EA-GNN). It uses a structure-aware aggregation with adaptive importance and an ego-awareness mechanism to help the model better utilize structural information of nodes' ego networks. This guards against noise interjected from a node's neighbourhood in the aggregation process.

The main contributions are listed below:

1) EA-GNN includes a structure-aware aggregation with adaptive importance by using a modified Jaccard similarity coefficient within the ego network of a node. This ego aggregation utilizes the second-order structural information to calculate weights.
2) EA-GNN uses an ego-awareness index, r, as the critical parameter that controls the balance between node features and graph structure. A greater value of r allows the model to focus more on node features, while a smaller r value makes the model place more weight on the graph structure. This allows EA-GNN to be more expressive than some of the existing GNN models (e.g., GCN, GAT, and GraphSAGE), which use the sum or mean aggregator.
3) The efficacy of EA-GNN is validated on eleven real-world networks through comparison to GCN, GAT, GraphSAGE, and non-local GNNs, along with MLP. The experimental results demonstrate that the proposed approach can reject noise introduced from graph structure, and that it maintains the performance for datasets with rich node features. On the other hand, it can also improve the performance in "typical" datasets where the graph structure is informative.
4) EA-GNN shows more robustness than its counterparts against network structure perturbations and node feature absence, making it more applicable to datasets with

different characteristics. Compared with its counterparts, EA-GNN is generic in the sense that it can reliably achieve better performance compared to MLP on not only assortative and feature-insufficient graphs, but also disassortative and feature-rich graphs.

II. Related Work

Graph Neural Networks (GNNs) serve as a general framework for constructing deep neural networks on graph data. GNNs can be categorized into two main streams: spectral-based and spatial-based approaches [17]. Spectral-based methods apply convolution operations in the spectral domain, achieved by computing the eigendecomposition of the Laplacian matrix of the input graph [45], [46], [47]. While these methods are computationally efficient, they heavily rely on the specific structure of the input graph, making models trained on one graph less adaptable to different graphs. In contrast, spatial-based approaches establish convolutions directly on the graph itself [2], [3], [48]. In this approach, node representations are updated iteratively by aggregating information from neighbouring nodes, allowing for the spatial propagation of node information throughout the graph. Compared to spectral approaches, spatial approaches receive more attention for their higher flexibility, generality, and efficiency [17].

Neural message passing is the key of spatial-based GNNs, which allows vector messages to be exchanged between nodes. It includes two steps: neighborhood aggregation and feature transformation. Most existing GNNs improve the task performance by designing better neighborhood aggregation operators. For example, GCN employs a weighted sum aggregator that assigns weights based on the degrees of neighboring nodes [1]. GAT is the first GNN model to apply an attention mechanism [3]. This mechanism computes attention weights for each neighbour, allowing for the weighting of each neighbour's influence during the aggregation process. GraphSAGE offers various aggregators such as mean aggregator, LSTM aggregator, and pooling aggregator, enabling adaptability to different datasets [2]. Deep GNNs, such as JKnet [49], GCNII [50], and RevGNNDeep [51], maintain the local information of node representations by incorporating the output from shallow layers into deeper layers, following a residual-style design approach.

Although GNNs have proven to be effective in a wide range of applications, their performance can deteriorate when confronted with unfavorable graph structures. For instance, conventional GNN models like GCN and GAT may exhibit reduced performance on graphs characterized by abundant node features and disassortative structures, which can introduce noise into the learning process. Several approaches have been developed to address this issue. GEOM-GCN uses a novel geometric aggregation scheme to capture long-range dependencies by computing the distance between every pair of nodes [43]. Due to its computational complexity, this model is challenging to run on large-scale graphs. Non-local GNN provides a simple yet effective aggregation framework, which applies attention-guided sorting and non-local aggregation to gather information from distant but informative nodes [44]. SimP-GCN introduces a feature similarity-preserving aggregation method to harmonize the information derived from both the graph structure and node features [52]. These methods can help to extract the informative message and mitigate noises from their neighbourhood to a certain degree, but they cannot handle graphs with rich features well, where neighbour interference is more likely to appear.

Recent research has been leveraging the structural information within local neighbourhoods to develop effective message-passing aggregation techniques, thereby enhancing the expressive capabilities of GNNs. For example, SHADOW-GNN involves the extraction of a localized subgraph, upon which a GNN of arbitrary depth is applied to bolster the GNN's expressive power [53]. NGNN, on the other hand, focuses on extracting a local subgraph surrounding each node and subsequently applies a base GNN to each subgraph to learn subgraph representations [54]. GraphSNN introduces a novel approach for injecting structural information into message-passing aggregation by establishing a new hierarchy of local isomorphism on neighbourhood subgraphs [55]. LAGNN incorporates a straightforward yet effective data augmentation strategy known as local augmentation. This strategy is designed to learn the distribution of node representations among neighbours based on the central node's representation, thereby enhancing GNN expressiveness with generated features [56]. These studies collectively highlight the pivotal role of local information in training GNN models and devising robust, high-performance GNNs.

III. EA-GNN: Ego-Aware Graph Neural Network

We present a generic Ego-Aware Graph Neural Network (EA-GNN). This proposed GNN introduces a local structural weighting mechanism, which computes amplification coefficients for local neighbouring messages based on the structure of the ego-network. This mechanism gives high importance to the neighbours that are *structurally similar* to the ego nodes, so it can balance the feature weights between ego nodes and their neighbours during message passing.

A. Structure-Aware Aggregation in Ego-Networks

Messages aggregation is essential for spatial GNNs [16]. Most previous studies [1], [2], [3] use simple aggregation functions. For example, GCN weights node features based on node degrees, GAT uses a self-attention mechanism to weight neighbourhood messages, and GraphSAGE adopts a mean aggregation to extract information from neighbourhood features. These aggregation schemes only use the simple structural information, such as local connection and node degree. However, there is more structural information that can be exploited for structure-aware aggregation. Here, we consider leveraging the structural similarity between neighbours of an ego node to modulate messages from them with different weights during the aggregation.

Structural similarity is a fundamental approach to measuring node similarity in a network. That is, two nodes are structurally similar if they are situated comparably in the network context [14]. There are a number of metrics of structural similarity, such as the number of common neighbours, cosine similarity,

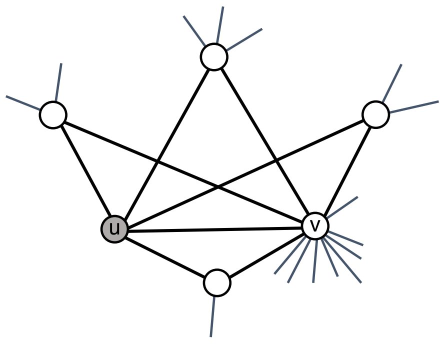


Fig. 2. Structural similarity of node u and node v when performing feature aggregation of node u.

Pearson correlation coefficient, and Jaccard similarity coefficient [57]. The number of common neighbours is the most simple similarity index, but it is too coarse and not normalized, making it difficult to compare the similarity of nodes with a massive difference in degrees. Cosine similarity and Pearson correlation coefficient are two efficient and widely used measures for capturing similarity between two sets of data. However, they are not very suitable for capturing structural similarity between set-based data, and their calculation load is relatively considerable. Cosine similarity is an angle-based measure and is particularly suitable for capturing similarity in the direction or orientation of vectors. The Pearson correlation coefficient is particularly useful for quantifying linear relationships.

Considering its efficiency, simplicity, and interpretability, EA-GNN uses a variant of the Jaccard similarity coefficient to capture the structural similarity. The Jaccard similarity is well-suited for this scenario, i.e., set-based comparison, where the order or frequency of elements in the sets does not matter. Given a graph $G = (V, E)$, the Jaccard similarity coefficient between nodes u and v is defined by

$$J_{uv} = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}, \tag{1}$$

where $N(u)$ and $N(v)$ are the neighbourhoods (i.e. set of neighbours) of u and v, and $|N(u)|$ and $|N(v)|$ are their degrees. In addition, the closed neighbourhood of given node u is denoted $N[u]$, i.e. $N[u] = N(u) \cup \{u\}$. In particular for J_{uv}, if nodes u and v share all the same neighbours, we have $N(u) = N(v)$. Thus, $|N(u)| = |N(v)| = |N(u) \cap N(v)| = |N(u) \cup N(v)|$, yielding $J_{uv} = 1$. When the two nodes have no common neighbours, i.e. $|N(u) \cap N(v)| = 0$, yielding $J_{uv} = 0$. Generally, $J_{uv} \in [0, 1]$ and it is symmetric.

Although the Jaccard similarity coefficient is a good index to measure the structural similarity between a node pair, it cannot be directly applied for re-weighting messages because the mutual influence between two nodes is often asymmetric. If a node has a small degree and a neighbour has a very large degree, the Jaccard similarity coefficient between them will always be small even if the neighbourhood of the low-degree node is mostly covered by that of the high-degree node. For example, consider neighbouring nodes u and v in the scenario in Fig. 2, where $N[u] \subset N[v]$ and $N[u] \ll N[v]$. As the model performs feature aggregation at u, we should acknowledge that v should influence u more than the other way round because v has a significantly more diverse neighbourhood than u. However, the calculated J_{uv} with (1) will be very small due to the high degree difference between node u and node v. Therefore, the Jaccard similarity coefficient cannot be directly used to measure the similarity between node and neighbours. To address this problem, we propose to calculate Jaccard similarity coefficient *within* the ego network of a given node because a node's ego network is more relevant with regard to the ego node than further structures of the network. In this way, the modified Jaccard similarity coefficient in node u's ego network is defined as

$$\overline{J}_{uv} = \frac{|\overline{N}(v)|}{|N(u)|}, \tag{2}$$

where $|\overline{N}(v)|$ is the degree of node v in node u's ego network. Note that $\overline{J}_{uv} \neq \overline{J}_{vu}$ here to signify the asymmetry of the mutual influence between u and v.

Our proposed aggregation scheme considers information flow within the ego network. It allows ego nodes to receive weighted information from their neighbours regulated by the local ego network structure. Specifically, the aggregated information for node u can be calculated as

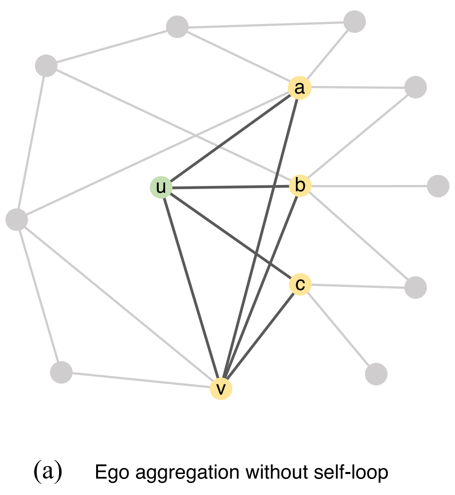
$$\begin{aligned}\mathbf{z}_u &= \frac{1}{\sum_{v \in N(u)} \overline{J}_{uv}} \sum_{v \in N(u)} \overline{J}_{uv} \mathbf{z}_v \\ &= \frac{1}{\sum_{v \in N(u)} |\overline{N}(v)|} \sum_{v \in N(u)} |\overline{N}(v)| \, \mathbf{z}_v,\end{aligned} \tag{3}$$

where $\mathbf{z}_u \in \mathbb{R}^d$ and $\mathbf{z}_v \in \mathbb{R}^d$ are the embedding of nodes u and v in a specific layer, respectively.

Fig. 3(a) shows an example of the proposed ego aggregation scheme. Consider that the model is performing information aggregation at node u. Here, node u's ego network includes the node set $\{u, v, a, b, c\}$. Then, the weight for each node can be calculated using its degree within the ego network. In particular, the weights for nodes $\langle u, v, a, b, c\rangle$ are $\langle \frac{4}{14}, \frac{4}{14}, \frac{2}{14}, \frac{2}{14}, \frac{2}{14}\rangle$, respectively. Note that node v has the same weight as the ego node u in this example, and this weight is the greatest in the ego network. This is reasonable because nodes v and u are directly joined and have similar local structures, i.e., they have the same common neighbours. In essence, this ego aggregation utilizes the second-order structural information to calculate weights for the neighbourhood messages within one aggregation step.

To give greater importance to the ego node than any of its neighbours, a self-loop to the ego node is added to ensure it always has the greatest weight. Fig. 3(b)) shows an example of this self-loop. In this case, the weights for nodes $\langle u, v, a, b, c\rangle$ become $\langle \frac{5}{15}, \frac{4}{15}, \frac{2}{15}, \frac{2}{15}, \frac{2}{15}\rangle$, respectively. As such, aggregator will pay more attention to the ego node than any other nodes within its neighbourhood. The aggregation function we adopt is

$$\mathbf{z}_u = \frac{1}{\sum_{v \in N[u]} |\overline{N}(v)|} \sum_{v \in N[u]} |\overline{N}(v)| \, \mathbf{z}_v, \tag{4}$$



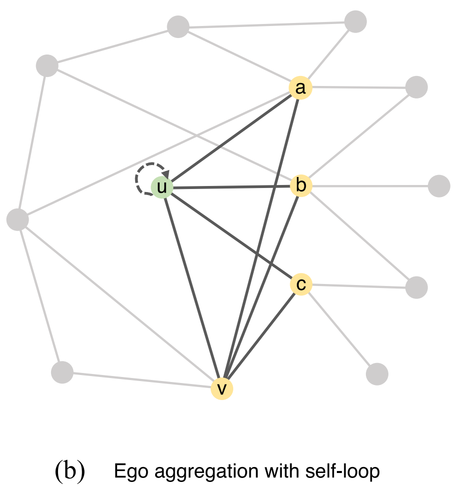


Fig. 3. Illustration of the ego aggregation scheme with adaptive importance.

where $N[u] = N(u) \cup \{u\}$. The relative contribution among neighbours can be further adjusted using a power function, as discussed next.

B. Ego Awareness

In graph learning, the network structure and node features are the two factors contributing to the performance of a machine learning task. However, when indiscriminately compounded, they do not always yield better performance than using node features alone. This is especially true for disassortative graphs, where neighbhouring nodes are often dissimilar. In this case, a node's neighbourhood often contains a considerable amount of misinformation with respect to the ego node, undermining the quality of the aggregated information. The richness of node features is also relevant here because a feature-rich node should be less reliant on information coming from neighbours. For networks with very rich node features, such as the two coauthorship networks (Fig. 1), even a simple MLP can achieve great accuracy. Somewhat surprisingly, the example GNNs do not perform very well on these datasets in the comparative experiments. This occurs because, during the aggregation in a GNN, the ego node is dependent on information from the neighbourhood. However, a node with rich features should be more confident in its own information, and the information from neighbours should be referred to with more caution. Therefore, in this case, the ego node needs a balance mechanism when node features and network structures compete with, rather than reinforce, each other.

An ego-aware aggregation is proposed within EA-GNN to strengthen the desired information and weaken the unexpected noises. This mechanism introduces a power function to the aggregation to flexibly adjust the contribution of each neighbour relative to the ego node. The aggregated feature at node u can be calculated by

$$\mathbf{z}_u = \frac{1}{\sum_{v \in N[u]} \left|\overline{N}(v)\right|^r} \sum_{v \in N[u]} \left|\overline{N}(v)\right|^r \mathbf{z}_v, \tag{5}$$

where the hyperparameter r is essentially an ego-awareness index that enables the model to guard against useless information. When $r = 0$, the model is similar to GraphSAGE because it gives the ego node and each neighbour the same importance. In this case, the model aggregates information from the neighbourhood with full trust so that the ego node's representation is heavily influenced by its neighbours, leading to a performance degradation for some disassortative networks or networks with rich node features. To avoid this problem, r can be increased to make the model focus more on its own features as well as those from structurally similar neighbours, with lesser importance on structurally dissimilar neighbours. In the extreme condition, with a large r, the model becomes similar to MLP in that no consideration is given to the graph structure. This hyperparameter r can be tuned via cross-validation.

C. Graph Convolution With Adaptive Weighting

A graph neural network could have multiple convolutional layers. The model generates node representations in each layer using its previous layer's node embeddings. Specifically, a convolutional layer in graph neural networks normally contains a feature aggregation step and an update step. The update of the l-th layer can be expressed as

$$\mathbf{h}_u^{(l)} = \mathbf{UPDATE}^{(l-1)}\Big(\mathbf{AGGREGATE}^{(l-1)} \Big(\{\mathbf{h}_v^{(l-1)}, \forall v \in N(u)\} \cup \mathbf{h}_u^{(l-1)}\Big)\Big), \tag{6}$$

where $\mathbf{h}_u^{(l)} \in \mathbb{R}^d$ is the representation at node u in the l-th layer. In each layer, the aggregation function takes the representations of the nodes in the neighbourhood $N(u)$ of node u as input and computes an aggregated representation from them. Then, the update function generates a new representation at node u with the aggregated representation and its own representation in the previous layer. Formally, the representation update equation is written as

$$\mathbf{h}_u^{(l)} = \sigma\left(\sum_{v\in N[u]} \mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)}\right), \tag{7}$$

where $\mathbf{W}^{(l)}$ is the learnable transformation matrix for the l-th layer, and σ is the activation function, e.g., ReLU. The re-weighting matrix $\mathbf{E}^r$ is introduced to incorporate structural information of ego networks, which has dimension the same as the adjacency matrix of a network. The elements of $\mathbf{E}^r$ are defined as

$$\mathbf{E}_{uv}^r = \frac{\left|\overline{N}(v)\right|^r}{\sum_{v\in N[u]}\left|\overline{N}(v)\right|^r}. \tag{8}$$

That is, the element $\mathbf{E}_{uv}^r$ is the reweighting coefficient indicating the importance of node v to node u within u's ego network adjusted by the exponent r. Note that $\mathbf{E}_{uv}^r \neq \mathbf{E}_{vu}^r$ even in an undirected network since nodes u and v may not have mutually symmetric influence. Therefore, the final representation update equation is

$$\mathbf{h}_u^{(l)} = \sigma\left(\sum_{v\in N[u]} \mathbf{E}_{uv}^r\mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)}\right). \tag{9}$$

D. Model Analysis

Both node features and graph structure are determinant factors of the performance of GNN models. For example, if node features are very rich, which means a model can already achieve a high performance using node features only, aggregated messages from neighbours may contain more noise than useful information. Thus, GNN models with message passing, such as GraphSAGE, hit a performance ceiling compared to a simple MLP in graphs with rich node features. Unfavourable graph structure is another problem with vanilla GNN models. In real-world applications, it is impossible to collect the entirely faithful interrelationship between nodes for intended or unintended reasons. For example, fraudsters pretend to connect to many benign users in financial networks. As a result, the graph indicating the relationship between nodes is not always accurately built. In such cases, the messages from neighbours also include interference that caps the performance of GNN models. The proposed EA-GNN model can solve this interference problem and maintain the advantages of vanilla GNNs by adjusting the ego-awareness index r. Here, a proof is presented that shows EA-GNN bridges the MLP model and GraphSAGE.

Starting with (8), when $r \to \infty$, we have

$$\begin{aligned}\mathbf{E}_{uv}^\infty &= \frac{\left|\overline{N}(v)\right|^\infty}{\sum_{v\in N[u]}\left|\overline{N}(v)\right|^\infty}\\ &= \begin{cases}0 \quad , \quad v\neq u,\\ 1 \quad , \quad v=u.\end{cases}\end{aligned} \tag{10}$$

Inserting (10) into (9) yields

$$\begin{aligned}\mathbf{h}_u^{(l)} &= \sigma\left(\sum_{v\in N[u]} \mathbf{E}_{uv}^\infty\mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)}\right)\\ &= \sigma\left(\sum_{v\in N(u)} \mathbf{E}_{uv}^\infty\mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)} + \mathbf{E}_{uu}^\infty\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}\right)\\ &= \sigma\left(\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}\right).\end{aligned} \tag{11}$$

Therefore, when $r\to\infty$, EA-GNN only utilizes node features without considering the graph structure, becoming equivalent to MLP.

Next, consider the case for EA-GNN when $r=0$. The re-weighting matrix $\mathbf{E}^r$ becomes

$$\begin{aligned}\mathbf{E}_{uv}^0 &= \frac{\left|\overline{N}(v)\right|^0}{\sum_{v\in N[u]}\left|\overline{N}(v)\right|^0}\\ &= \frac{1}{\left|\overline{N}(u)\right|}.\end{aligned} \tag{12}$$

Substituting (12) into (9) yields

$$\begin{aligned}\mathbf{h}_u^{(l)} &= \sigma\left(\sum_{v\in N[u]} \mathbf{E}_{uv}^r\mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)}\right)\\ &= \sigma\left(\sum_{v\in N[u]} \frac{1}{\left|\overline{N}(u)\right|}\mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)}\right).\end{aligned} \tag{13}$$

Equation (13) shows that for EA-GNN with $r=0$, each neighbour has the same weight $\frac{1}{|\overline{N}(u)|}$ in the node embedding update step, which is equivalent to GraphSAGE.

Therefore, MLP and GraphSAGE are two special cases of EA-GNN. EA-GNN is a generic GNN model that can be tuned to sit between MLP and vanilla GNNs, such as GraphSAGE, through adjusting the ego-awareness index, r. As a result, EA-GNN can adapt to datasets with different characteristics. For example, EA-GNN can increase r when running on the datasets with very rich features, emulating MLP to make full use of node features. On the other hand, when the graph is well built and node features are not informative enough, decreasing index r will make the model pay more attention to the messages from the neighbours.

E. Time Complexity Analysis

The additive cost of EA-GNN in comparison to conventional GNNs like GCN and GAT arises from the computation of the reweighting matrix $\mathbf{E}^r$ (8). The time complexity of computing reweighting elements for a single node may be expressed as $O(\overline{k})$ where $\overline{k}$ is the average degree of the network since the degree of each neighbour of this node needs to be calculated in its ego network. Then, the time complexity of computing reweighting elements for all nodes can be $O(|N|\overline{k})$, where $|N|$ is the node number of the network. Since $|E| = \frac{|N|\overline{k}}{2}$, the computational complexity of calculating the reweighting matrix

TABLE I
STATISTICS OF THE DATASETS USED IN OUR EXPERIMENTS

Dataset	Cora	CiteSeer	PubMed	Amazon_C	Amazon_P	Coauthor_C	Coauthor_P	Cornell	Wisconsin	Texas	Actor
#Nodes	2708	3327	19717	13381	7487	18333	34493	183	251	183	7600
#Edges	5429	4732	44338	245778	119043	81894	247962	295	499	309	33544
#Features	1433	3703	500	767	745	6805	8415	1703	1703	1703	931
#Classes	7	6	3	10	8	15	5	5	5	5	5
$H(\mathcal{G})$	0.83	0.71	0.79	0.79	0.84	0.83	0.92	0.11	0.16	0.06	0.24

is $O(|E|)$, where $|E|$ is the edge number of the network. The calculation of the reweighting matrix only needs to be done once during the training process as part of data preprocessing. It has been reported that the time complexity of the typical GNNs, such as GCN and GAT, is $O(|E|)$[1], [3]. Therefore, the time complexity of the proposed EA-GNN is $O(|E|) + O(|E|)$, which is also $O(|E|)$. Although determining the optimal r for a specific dataset by cross validation requires extra time, this complexity is still on par with the popular methods.

IV. EXPERIMENTS

The effectiveness of EA-GNN is evaluated on a variety of open graph datasets. In particular, the aim is to confirm that EA-GNN can improve performance and adaptability compared to MLP, GCN, GAT, GraphSAGE, and non-local GNNs.

A. Datasets

The proposed EA-GNN is evaluated using eleven open graph datasets. Table I provides a summary of the statistics for these datasets. The homophily metric can be an effective method for distinguishing assortative and disassortative graph datasets. The homophily of a graph $\mathcal{G}$ is defined as

$$H(\mathcal{G}) = \frac{1}{|V|}\sum_{v\in V}\frac{|\{u : u \in \mathcal{N}(v) \text{ and } l(u) = l(v)\}|}{|\mathcal{N}(v)|}, \quad (14)$$

where $|\{u : u \in \mathcal{N}(v) \text{ and } l(u) = l(v)\}|$ represents the count of nodes directly connected to node v that share the same label as node v and $\mathcal{N}(v)$ denotes the set of node v's direct neighbours [43]. Cora, CiteSeer, and PubMed are well-established benchmark citation network datasets [58]. In these networks, nodes represent research papers, while edges are citation relationships between the papers. Node features in this context represent the bag-of-words representations of papers, while the node labels signify the topics associated with each paper. Amazon Computer and Amazon Photo are subsets of the Amazon co-purchase network [59]. Within these datasets, nodes represent products, and the presence of edges between two products implies a frequent co-purchasing pattern. Node features are derived from the bag-of-words representations of product reviews, and the node label indicates a product's category. Coauthor CS and Coauthor Physics are two co-authorship networks sourced from the Microsoft Academic Graph, originally used in the KDD Cup 2016 challenge [59]. In these networks, authors are represented as nodes, and an edge between two authors signifies their collaboration on a research paper. Node features are derived from the keywords associated with an author's papers, while the node label categorizes the author's primary area of research expertise. Cornell, Texas, and Wisconsin datasets are part of the WebKB dataset collection gathered by Carnegie Mellon University [43]. Within these datasets, nodes represent web pages, and edges signify hyperlinks between them. Node features are derived from the bag-of-words representations of the web pages, and node labels encompass categories such as student, project, course, staff, and faculty. The Actor dataset constitutes an actor relation network, where nodes correspond to actors and edges denote their co-occurrence on the same web page sourced from Wikipedia [60]. Node features are constructed from the bag-of-words extracted from keywords found in the actors' Wikipedia pages. The node labels encompass five categories derived from the topics of the actors' Wikipedia pages.

B. Experimental Settings

To demonstrate the effectiveness of the proposed model, EA-GNN is compared with MLP, GCN, GAT, GraphSAGE, non-local GCN, and non-local GAT on inductive node label classification tasks. MLP, GCN, GAT, GraphSAGE, and the proposed EA-GNN are implemented using Pytorch [61] and Pytorch Geometric [62]. For the non-local GCN and non-local GAT, an implementation provided by the authors [44] is used. All the latent representations are 128-dimension. The activation function σ is ReLU, and the Adam optimizer with a learning rate 0.005 is used to minimize the cross entropy losses. Note that the index r, which controls the ego-awareness mechanism, is distinct between different datasets as it is determined by a hyper-parameter search for each model using the validation set. An early stopping policy is used that is based on the validation accuracy within 500 epochs, with the patience set to 5. All the experiments are conducted on an Ubuntu 20.04.4 LTS system with Lenovo ThinkStation, Xeon 5118, 256 GB RAM and NVIDIA RTX 2080 Ti with 12 GB memory size. The software used for the experiments is Python 3.8.5, PyTorch 1.10.0, PyG 2.0.2, NumPy 1.19.1, CUDA 11.3.1, and CUDNN 8.2.0.

For all graph datasets, the nodes are randomly split into 60%, 20%, and 20% for the training, validation, and testing sets. In the training phase, only nodes in the training set and edges between the training nodes are used to train the model. The edges between training nodes and validation nodes or testing nodes are omitted. In the validation and testing phases, new nodes from the validation set or testing set are added to the existing network. The model then utilizes the existing network in combination with

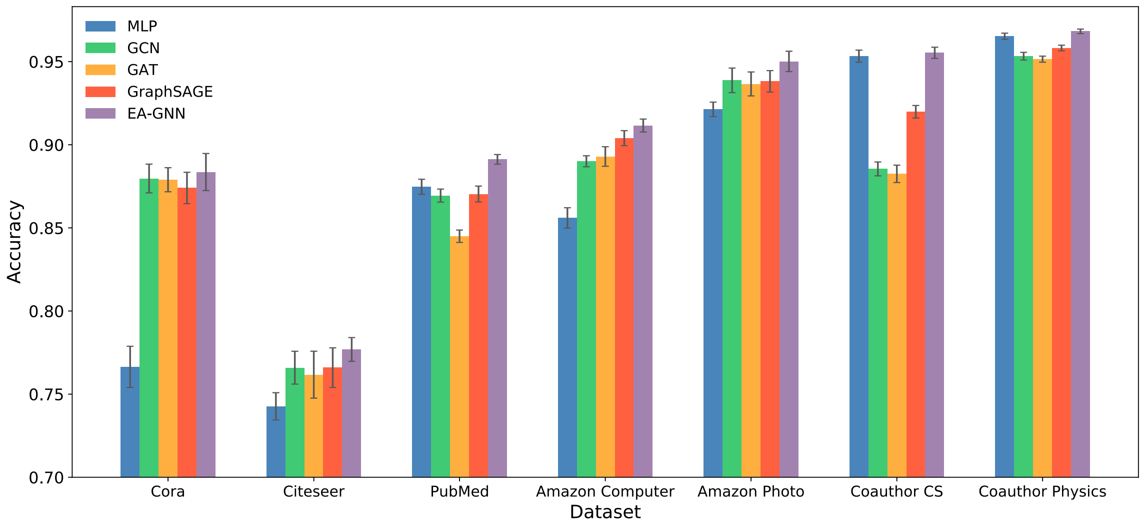


Fig. 4. Accuracy comparison of different models on different datasets.

new nodes' features and connection information to predict new nodes' classes. This experiment mimics the evolving nature of information networks in reality. Networks grow with introduction of new unseen nodes, and often these new nodes need to be classified. Ten different random splits with the same training, validation, and testing sizes are used to mitigate the effect of different splits on the performance.

C. Results

Four experiments are conducted to demonstrate EA-GNN's effectiveness. First, a comparison is made of the classification accuracy between EA-GNN and other existing GNNs, as well as MLP. Second, it is demonstrated that the optimal r is different for different datasets by cross validation. The third and fourth experiments test the robustness of EA-GNN and other baselines by rewiring parts of the networks and ignoring subsets of node feature dimensions.

1) Comparison of Classification Accuracy Among ML Methods: EA-GNN is compared to popular GNN models to show the general effectiveness of the proposed model, in addition to MLP, which only uses on the node features.

Fig. 4 shows the classification accuracy comparisons of different models on seven assortative datasets. It can be observed that EA-GNN outperforms all other models on the seven datasets. It is worth noting that MLP outperforms some of the existing GNN models, such as GraphSAGE and GAT, especially on the PubMed, Coauthor CS, and Coauthor Physics datasets. That occurs because the information aggregated from neighbourhoods in these three networks contributes negatively to the performance of the GNN models.

There are several reasons that may lead to the loss of performance. First, the network is feature-rich, which means node features are abundant and of high quality so that label prediction using only node features can achieve relatively high accuracy. In these kind of networks, node features are strong indicators of node labels. The aggregated information from neighbourhood will conversely interfere with the model's judgement. The other reason that may cause this problem is that the network is not well-built or too noisy so that the message passing mechanisms are unreliable. Thus, these reasons lead to the same effect: the aggregated neighbourhood information brings more interference than utility. Our proposed EA-GNN method, however, can effectively avoid this neighbourhood interference problem.

The accuracy of our approach is consistently the highest even for the three feature-rich datasets. Fig. 4 presents visual comparisons of these methods, while Table II provides a summary of the detailed classification accuracy of the models. It can be noticed that for the four disassortative datasets (i.e., Cornell, Wisconsin, Texas, and Actor), EA-GNN performs better than other GNNs, including NLGCN and NLGAT, even though its accuracy is slightly lower than MLP on Cornell, Wisconsin, and Texas. A possible explanation is that these three datasets are small-scale, only containing about 200 nodes and less than 500 edges, which creates uncertainty in the experiment. For the normal-scale Actor dataset, EA-GNN still outperforms all other models though it is disassortative.

2) Changes of Classification Accuracy With Ego-Awareness Index r: Fig. 5 shows how the class prediction accuracy changes with different index r on different datasets. All the experiments are repeated 10 times. Shaded regions denote the standard deviation. Grid search is used to find the optimal value of r that yields the model with the highest accuracy for each dataset.

Recall that r is an exponent that can adjust the contribution of each neighbour in the local aggregation step. Higher r means the model pays more attention to high degree nodes in the ego network. Note that the ego node always holds the highest degree in the ego network. Therefore, as the value of r approaches 0, the distinctions in importance among the ego node's neighbours diminish, causing the model to exhibit a behavior akin to that of GraphSAGE. With a large enough r, the model will assign the ego node with the highest degree in its ego network the largest weight, and the neighbours' weights can be ignored to some degree. In this way, the model degenerates to MLP. By

TABLE II
Mean Classification Accuracy on Benchmark Datasets

Dataset	MLP	GCN	GAT	GraphSAGE	NLGCN	NLGAT	EA-GNN
Cora	76.6±1.2	88.0±0.9	87.9±0.7	86.9±1.2	84.4±2.4	87.6±1.1	**88.1±1.1**
CiteSeer	74.2±0.8	76.6±1.0	76.2±1.4	76.7±0.9	72.1±2.2	75.4±1.6	**77.7±0.9**
PubMed	87.5±0.5	87.0±0.4	84.5±0.4	87.0±0.6	87.2±0.4	87.0±0.4	**89.2±0.3**
Amazon Computer	85.6±0.6	89.0±0.3	89.3±0.6	89.8±0.4	81.7±7.9	89.5±0.6	**90.8±0.3**
Amazon Photo	92.1±0.4	93.9±0.7	93.7±0.7	93.5±0.5	87.0±8.1	93.5±0.5	**94.7±0.6**
Coauthor CS	95.3±0.4	88.5±0.4	88.3±0.5	91.6±0.5	92.7±0.5	92.8±0.3	**95.5±0.3**
Coauthor Physics	96.5±0.2	95.3±0.2	95.1±0.2	95.5±0.2	96.4±0.2	96.3±0.2	**96.7±0.2**
Cornell	**81.6±6.3**	47.0±5.6	45.9±5.0	41.5±7.5	46.2±6.6	44.1±6.6	76.5±4.5
Wisconsin	**84.9±5.3**	54.3±5.0	50.0±5.6	48.2±5.3	50.2±3.8	49.0±6.7	79.7±4.6
Texas	**81.3±7.1**	65.1±9.1	58.6±5.9	55.1±9.0	59.5±6.9	60.3±7.5	76.2±9.0
Actor	35.1±0.8	30.9±0.9	26.8±1.5	28.7±1.1	28.8±2.7	28.6±2.1	**36.0±1.0**

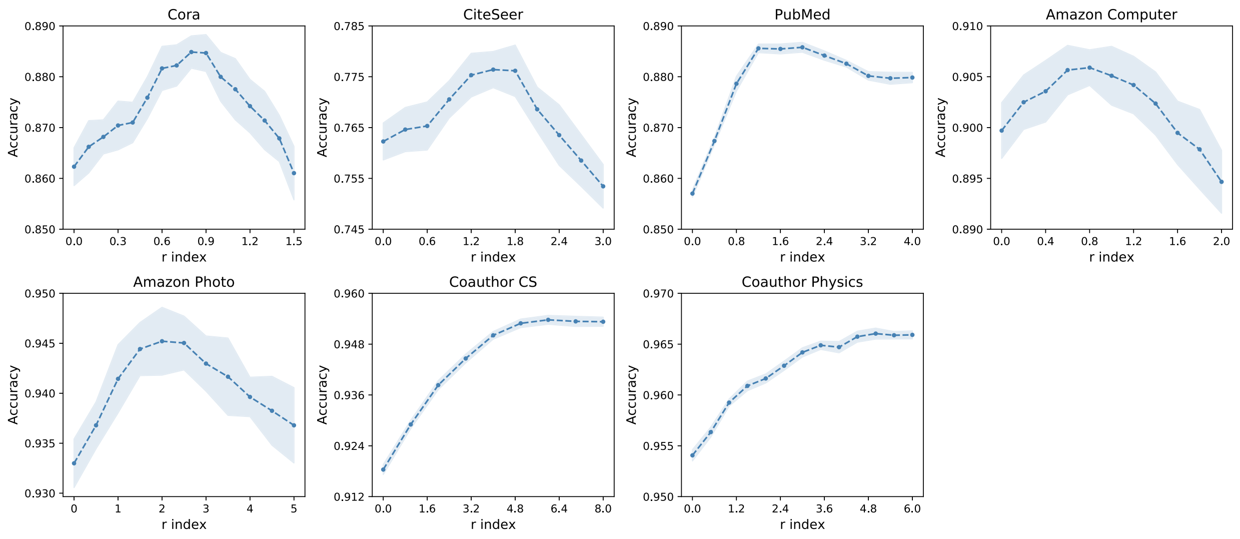


Fig. 5. Trend of accuracy with the change of r in different datasets.

adjusting r, EA-GNN has a high tolerance of datasets with different characteristics, making it widely applicable.

Fig. 5 shows that the curves for the Cora, CiteSeer, Amazon Computer, and Amazon Photo datasets have the same approximate trend – the accuracy at first increases with increasing r, then reaches a peak and subsequently decreases. In these four datasets, when r equals 0, corresponding to the GraghSAGE, the accuracy is relatively low. The reason is that the neighbourhood provides both useful information and interference. With the growth of r, the model pays more attention to the high degree nodes and less attention to the low degree nodes in the ego networks. This process can be viewed as an information distillation. Finding the best r means the model find a balance between nodes' own features and their neighbourhood influence. After that, continuing increasing r can help reject useless information, but the model also ignores useful information from the neighbourhood. That is why the model shows a decreasing accuracy.

The curve in the PubMed dataset is similar to those in the previous four datasets, with the only difference that the accuracy decreases to a stable level after the peak. This means the model degrades to MLP when r exceeds 3.5. The curves in the two Coauthor datasets do not experience a noticeable decrease. This is caused by the characteristics of the datasets, where nodes' neighbours provide little useful information and node-specific features dominate the prediction task. Overall, this experiment demonstrates that our model is widely applicable and can achieve a better performance regardless of whether the network is dominated by node-specific features or needs more information from nodes' neighbourhood.

3) Robustness Against Perturbations of Network Structure: In the real world, it is impossible to capture the entire network structure accurately due to intentional or unintentional reasons. For example, fraudsters can make some connections with high-credit accounts to avoid detection in credit card fraud detection, or edges in the network might be added or lost by mistake

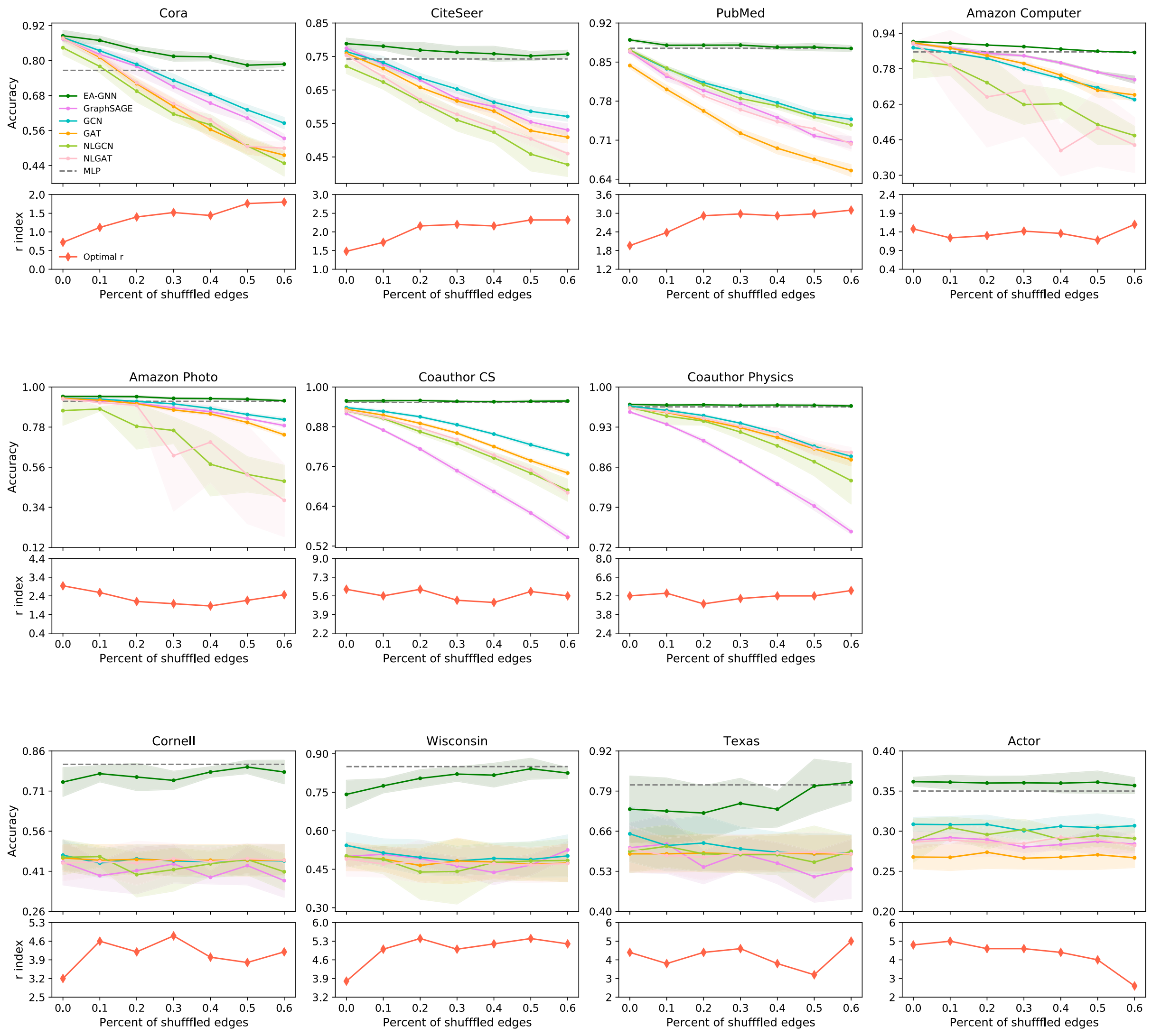


Fig. 6. Trend of accuracy by different approaches with the increasing ratio of shuffled edges on eleven datasets.

during the the data collection process. Therefore, the model's robustness against perturbations of the network structure is essential to avoid severe consequences in critical applications.

In this experiment, the node classification accuracy of different methods is evaluated with respect to perturbations of the network structure. These perturbations are simulated by shuffling a fraction of edges of the network. The edge shuffling replaces two randomly chosen edges u-v and x-y with the new edges u-x and v-y, so that noise is introduced to the network structure. Specifically, the ratio of shuffled edges is varied between 0 to 60% with a step of 10% for all the datasets. All the experiments are repeated 10 times with different random seeds for dataset split.

Fig. 6 shows the trend of accuracy by different models with the increasing ratio of shuffled edges on datasets. MLP ignores all relational information, thus the edge shuffling perturbations are irrelevant. The first seven datasets are assortative and the last four are disassortative. It can be observed that the classification performance of EA-GNN and the baselines drop at different rates with respect to the increasing ratio of shuffled edges for all of the assortative datasets. Specifically, the performance degradation rate of the baselines is much higher than EA-GNN. For example, in the Cora dataset, the performance gap between GraphSAGE and EA-GNN becomes more significant with the increasing shuffled edges. In the original unshuffled dataset, EA-GNN outperforms GraphSAGE by 1%, but the gap becomes about 16% when we randomly shuffle 40% edges. This demonstrates that EA-GNN is more robust to the change of edges. It should also be noted that the EA-GNN accuracy converges to the MLP accuracy when the network structure changes considerably. In contrast, the baseline accuracy decreases to be lower than the MLP accuracy. This convergence property allows EA-GNN to be applied with confidence, without worry that the graph structure quality will deteriorate the performance below MLP. For the disassortative datasets, both EA-GNN and the baselines do not show a noticeable performance change with the increase of shuffled edges, but EA-GNN still performs better than the baselines.

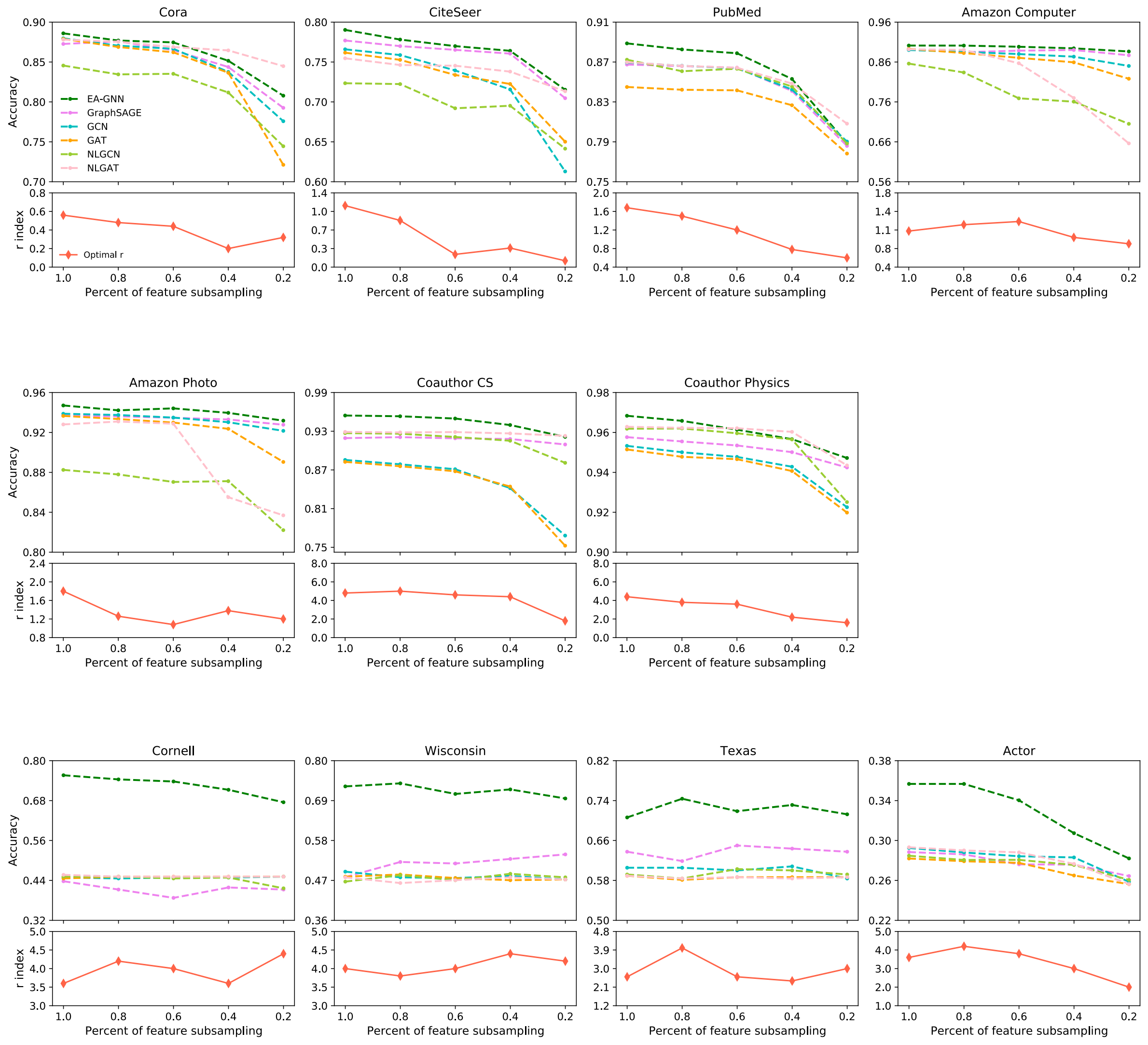


Fig. 7. Trend of accuracy of different approaches with the descending ratio of node feature sampling on eleven datasets.

The red line in Fig. 6 shows the trend of the optimal r with the increasing shuffled edges. It can be observed that the optimal r increases with the growth of shuffled edges in Cora, CiteSeer, and PubMed datasets. This occurs because increasing the ratio of shuffled edges introduces more disturbance to the network structure, leading to more noise in the aggregation step of GNNs. To mitigate the impact of noises from nodes' neighbourhood, the optimal r can be increased to make the model pay more attention to nodes' own features. During this process, the EA-GNN model approaches MLP (11). Therefore, as the ratio of shuffed edges increases, the performance curves of EA-GNN converges to that of MLP in all datasets in Fig. 6. This property of EA-GNN guarantees that EA-GNN performs better than MLP on any dataset regardless whether the graph structure is well built or not. Therefore, EA-GNN can solve the negative aggregation problem that the vanilla GNN encounters.

4) Robustness Against Missing Node Feature Dimensions: The richness of node features is another factor besides the graph structure that can influence the performance of the GNN models. To investigate the robustness of the proposed model to different richness levels of node features, different ratios of node feature dimensions are sampled from 100% down to 20% for each dataset, with a step size of 20%. The graph structure remains the same. In this experiment, EA-GNN and the baseline models are trained and tested for the node classification task. All the experiments are repeated 10 times with different random seeds for dataset split.

Fig. 7 shows the trend of accuracy of different approaches with the descending ratio of node feature sampling. The first seven datasets are assortative and the last four are disassortative. It is found that all methods experience performance degradation with the decrease of feature dimensions sampling, but the proposed EA-GNN sees a slower performance degradation in each dataset. For example, with the feature sampling ratio declining from 100% to 20% in Cora dataset, the accuracy of EA-GNN decreases from 88.1% to 80.8%, while the accuracy of GraphSAGE decreases from 86.9% to 79.25%. This performance degradation is reasonable since the models cannot acquire enough

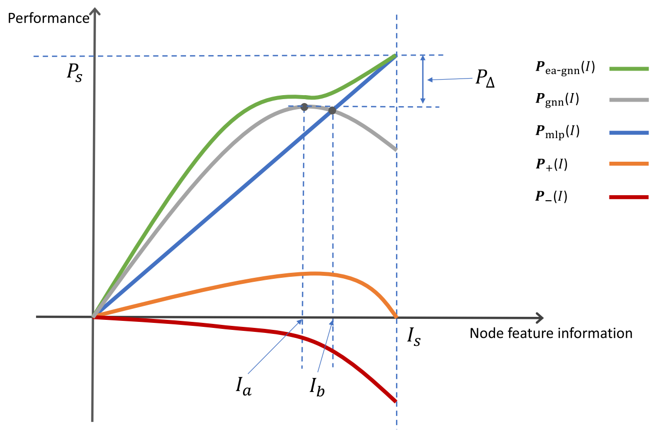


Fig. 8. Trend of $P_{\text{ea-gnn}}$, P_{gnn}, P_{mlp}, P_+, and P_- with the increase of I.

information from the data to perform the task with insufficient node features. However, our method performs better than the baselines in all the cases, thus demonstrating a stronger robustness against node features partially missing. The red line refers to the change of the optimal r of EA-GNN. Its overall trend is declining with the decreasing of feature sampling ratio because with less node feature information, EA-GNN is able to improve the importance of nodes' neighbours to aggregate more useful information from the neighbourhood. When optimal r goes down to close to 0, the performance of EA-GNN is close to that of GraphSAGE. That is, when r equals 0, EA-GNN behaves more like GraphSAGE.

V. Discussion

Node features and graph structure are two factors that can influence the performance of GNNs, and they usually reinforce each other. Generally speaking, GNN models can effectively improve performance on datasets where node features are moderate and similar nodes tend to connect to each other. However, as demonstrated in this work, when node features are very rich or graph structure is unfavourable, GNN models do not necessarily yield a significant performance gain compared to the MLP model, which only leverages node features. Rich node features often contain sufficient information for the model to make a fairly good decision, leaving little room for GNN models to further distill useful information from nearby nodes. If a graph is not well built, for example, when most nodes connect to nodes with different class labels, the aggregated information from neighbourhood may even interfere with the ego node's information representations, leading to a performance loss.

Fig. 8 illustrates how the richness of node features affect the performance gain of GNN models. Consider the situation of a static graph structure with no features, The x-axis in Fig. 8 reflects the amount of information carried by node features as new features are added to the graph. Here, the "amount of information" is governed by both the quantity and quality of node features. The y-axis is the estimated performance of MLP, GNN, and EA-GNN in a machine learning task at each point of feature richness. Given a reasonably strong MLP, its performance can be used as a reliable indicator of the node feature richness. For MLP to achieve a perfect performance, it requires a certain level of node feature richness, denoted I_s for *saturated information*. Therefore, MLP has a performance curve of the 45-degree line in $[0, I_s]$. In comparison, the aggregation process in a GNN allows an ego node to gather information from its neighbours to combine with its own. The information collected from neighbours, however, can include both positive, useful information and what turns out to be noisy. These two forces both contribute to the performance of GNN but cancel each other. In essence, the performance of GNN, denoted $P_{\text{gnn}}(I)$ as a function of I, is a compound of three factors:

$$P_{\text{gnn}}(I) = P_{\text{mlp}}(I) + P_+(I) + P_-(I). \tag{15}$$

where $P_+(I)$ is the performance contributed by the useful information from neighbours, and $P_-(I)$ is the performance loss due to neighbour interference.

As illustrated in Fig. 8, $P_+(I)$ increases with I when I is relatively small and decreases in a high-I region. This is because when I is small, which means node features are not informative, GNN can help nodes acquire useful information from their neighbourhood, so the room for performance improvement is large. As I increases towards near I_s, i.e. features are informative, the room for such a performance gain becomes less. On the other hand, $|P_-(I)|$ grows monotonically with I because a large I means node features already contain a great deal of useful information for the task, rendering the information gathered from neighbours useless. An extreme case is when the node features are saturated, i.e. $I = I_s$, none of the gathered information could help the model to do better. Thus, the performance curve of P_{gnn} has a peak when it makes the best use of neighbours information, denoted $I = I_a$. Note for the major of the I domain, GNN outforms MLP because the node features and network structure are synergetic. However, in a high I region, P_{gnn} drops below P_{mlp}, and we denote the crossover point I_b.

The region to the right of I_b was what our preliminary experiments demonstrated for the feature-rich datasets (Fig. 1). That is, the domain of $[0, I_s]$ is divided three regions by I_a and I_b. In the first region, where $I < I_a$, adding node features helps GNN perform better. Thus, GNN always outperforms MLP, i.e. $P_{\text{gnn}}(I) > P_{\text{mlp}}(I)$. In the second region, where $I_a < I < I_b$, $P_{\text{gnn}}(I)$ declines with the increase of I but is still greater than $P_{\text{mlp}}(I)$. Thus, GNN still performs better than MLP in this region although the improvement becomes less with the increase of I. There is also a gap, P_Δ, between $P_{\text{gnn}}(I_a)$ and the saturated information, P_s. This gap is determined by the graph structure quality and the design of GNN model. In the third region, where $I > I_b$, we have $|P_+(I_b)| < |P_-(I_b)|$, resulting in $P_{\text{gnn}}(I) < P_{\text{mlp}}(I)$, where GNN underperforms MLP. In this case, when node features are extremely rich, a naïve application of GNN models can lead to a lower performance than MLP due to the acquired negative information outweighting useful information. To address this issue, the proposed EA-GNN judiciously shields the neighbour interference when the node features are rich. In reference to (8), EA-GNN can balance the importance between the ego node and its neighbours through changing the ego-awareness index r. For example, when the node features are not rich, such as the Cora and CiteSeer datasets, EA-GNN can use a relatively small r to utilize more useful information from neighbours, as shown in Fig. 5. When the node features are very rich, such as the two Coauthor datasets, EA-GNN can apply a relative large r to resist noisy information from neighbours. In general, EA-GNN outperforms the better of GNN and MLP in the entire domain of $[0, I_s]$ and monotonically increases with I thanks to its ability in resisting the noisy information in the aggregation. As a result, on one hand at the low-I region, it does better than GNN while in the high-I region, its performance does not drop below that of MLP. In summary, node feature richness should be taken into consideration to avoid performance degradation when designing or applying GNN models.

VI. Conclusion

We have proposed a novel GNN leveraging structural information of nodes' ego networks called Ego-Aware Graph Neural Network (EA-GNN). This model paves the way for incorporating ego networks' structural information to the learned graph representations, which helps GNNs achieve a stable and greater performance. We argue that existing GNNs are somewhat specialized. They can achieve a considerable performance gain with graphs with moderate node features and assortative graphs. However, for graphs with rich node features and disassortative graphs, existing GNNs often do not introduce noticeable improvement, or even experience a performance loss when compared to MLP. The proposed EA-GNN can solve this problem by using a structure-aware aggregation with adaptive importance and an ego-awareness mechanism to help the model better utilize structural information of nodes' ego networks. This guards against noise interjected from a node's neighbourhood in the aggregation process. Therefore, EA-GNN can reliably achieve better performance compared to MLP on not only assortative and feature-insufficient graphs, but also disassortative and feature-rich graphs.

We anticipate that EA-GNN could be generalized to other machine learning tasks in graphs, such as link prediction and graph classification. The core function of GNNs is to generate representation vectors accurately. A GNN model performs well in node classification, meaning it is able to generate high-quality representation vectors. As a result, it is also expected to perform well in link prediction and graph classification. Thus, the proposed method can be used to improve the task performance by learning more accurate representations of nodes and graphs in many other fields, such as predicting cancer genes [35], detecting financial frauds [36], [37], and solving combinatorial optimization problems in graphs [38]. In the proposed model, the ego-awareness index, denoted as r, is regarded as a hyperparameter whose value is determined through cross-validation. As a result, the artificially selected value of r might not comprehensively capture the essence of adaptation characteristics. Therefore, one potential area of improvement would be building r into the graph neural network itself rather than having it as a hyperparameter to enable the model to capture the intrinsic adaptation essence and avoid the extra computation cost of cross validation.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2016.

[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.

[4] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Stat. Mechanics Appl.*, vol. 390, no. 6, pp. 1150–1170, 2011.

[5] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2071–2080.

[6] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining Deep Learn. Day*, 2018, pp. 1–7.

[7] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 974–983.

[8] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5165–5175.

[9] S. Pan, J. Wu, X. Zhu, C. Zhang, and S. Y. Philip, "Joint structure feature exploration and regularization for multi-task graph classification," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 715–728, Mar. 2016.

[10] S. Pan, J. Wu, X. Zhu, G. Long, and C. Zhang, "Task sensitive feature exploration and learning for multitask graph classification," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 744–758, Mar. 2017.

[11] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1666–1674.

[12] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4438–4445.

[13] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A fair comparison of graph neural networks for graph classification," in *Proc. Int. Conf. Learn. Representations*, 2020.

[14] M. Newman, *Networks*. London, U.K.: Oxford Univ. Press, 2018.

[15] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[16] W. L. Hamilton, "Graph representation learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 14, no. 3, pp. 1–159, 2020.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[18] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[19] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Netw.*, vol. 129, pp. 203–221, 2020.

[20] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[21] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook Brain Theory Neural Networks*, Cambridge, MA, USA: MIT Press, vol. 3361, pp. 255–258, 1998.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 84–90.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, 2010.

[25] M. H. Chehreghani, "Half a decade of graph convolutional networks," *Nature Mach. Intell.*, vol. 4, no. 3, pp. 192–193, 2022.

[26] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Representations*, 2018.

[27] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3634–3640.

[28] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 922–929.

[29] J. Xue et al., "Quantifying the spatial homogeneity of urban road networks via graph neural networks," *Nature Mach. Intell.*, vol. 4, no. 3, pp. 246–257, 2022.
[30] J. Jiménez-Luna, F. Grisoni, and G. Schneider, "Drug discovery with explainable artificial intelligence," *Nature Mach. Intell.*, vol. 2, no. 10, pp. 573–584, 2020.
[31] D. Duvenaud et al., "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.
[32] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: Moving beyond fingerprints," *J. Comput.-Aided Mol. Des.*, vol. 30, no. 8, pp. 595–608, 2016.
[33] X. Fang et al., "Geometry-enhanced molecular representation learning for property prediction," *Nature Mach. Intell.*, vol. 4, no. 2, pp. 127–134, 2022.
[34] Y. Wang, J. Wang, Z. Cao, and A. B. Farimani, "Molecular contrastive learning of representations via graph neural networks," *Nature Mach. Intell.*, vol. 4, no. 3, pp. 279–287, 2022.
[35] R. Schulte-Sasse, S. Budach, D. Hnisz, and A. Marsico, "Integration of multiomics data with graph convolutional networks to identify new cancer genes and their associated molecular mechanisms," *Nature Mach. Intell.*, vol. 3, no. 6, pp. 513–526, 2021.
[36] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 315–324.
[37] D. Wang et al., "A semi-supervised graph attentive network for financial fraud detection," in *Proc. IEEE Int. Conf. Data Mining*, 2019, pp. 598–607.
[38] Y. Peng, B. Choi, and J. Xu, "Graph learning for combinatorial optimization: A survey of state-of-the-art," *Data Sci. Eng.*, vol. 6, no. 2, pp. 119–141, 2021.
[39] C. Fan, L. Zeng, Y. Sun, and Y.-Y. Liu, "Finding key players in complex networks through deep reinforcement learning," *Nature Mach. Intell.*, vol. 2, no. 6, pp. 317–324, 2020.
[40] H. Chen, W. Qiu, H.-C. Ou, B. An, and M. Tambe, "Contingency-aware influence maximization: A reinforcement learning approach," in *Proc. Conf. Uncertainty Artif. Intell.*, 2021, pp. 1535–1545.
[41] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 5057–5069, Sep. 2022.
[42] W. Kool, H. V. Hoof, and M. Welling, "Attention, learn to solve routing problems," in *Proc. Int. Conf. Learn. Representations*, 2018.
[43] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2019.
[44] M. Liu, Z. Wang, and S. Ji, "Non-local graph neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 10270–10276, Dec. 2022.
[45] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Representations*, 2014.
[46] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[47] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, Jan. 2019.
[48] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *Proc. Int. Conf. Learn. Representations*, 2018.
[49] 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.*, 2018, pp. 5453–5462.
[50] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1725–1735.
[51] G. Li, M. Müller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6437–6449.
[52] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 148–156.
[53] H. Zeng et al., "Decoupling the depth and scope of graph neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2021, pp. 19665–19679.
[54] M. Zhang and P. Li, "Nested graph neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2021, pp. 15734–15747.
[55] A. Wijesinghe and Q. Wang, "A new perspective on 'how graph neural networks go beyond Weisfeiler-Lehman?'," in *Proc. Int. Conf. Learn. Representations*, 2022.
[56] S. Liu et al., "Local augmentation for graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 14054–14072.
[57] P. Jaccard, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.
[58] 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, pp. 93–106, 2008.
[59] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop,*, 2018, pp. 1–11.
[60] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2009, pp. 807–816.
[61] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Annu. Conf. Inf. Process. Syst.*, 2019, pp. 8024–8035.
[62] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," in *Proc. Int. Conf. Learn. Representations*, 2019.

Zhihao Dong (Student Member, IEEE) received the B.Eng. and M.Eng. degrees from Chongqing University, Chongqing, China, in 2010 and 2017, respectively. He is currently working toward the Ph.D. degree with Queen's University, Kingston, ON, Canada. His research interests include the intersection of machine learning and network science, network spreading process, and social network analysis.

Yuanzhu Chen (Member, IEEE) received the B.Sc. degree from Peking University, Beijing, China, in 1999, and the Ph.D. degree from Simon Fraser University, Burnaby, BC, Canada, in 2004. Since 2005, he has been a Professor of computing science, and is currently affiliated with the School of Computing, Queen's University, Kingston, ON, Canada. From 2004 to 2005, he was a Postdoctoral Researcher with Simon Fraser University. In 2005, he joined Memorial University, St. John's, NL, Canada, as a tenure-track Assistant Professor. While with Memorial, he was the Deputy Head for Undergraduate Studies from 2012 to 2015, Deputy Head for Graduate Studies from 2016 to 2019, and Department Head from 2019 to 2021. In 2021, he then joined the Queen's School of Computing. His research interests include complex networks, computer networking, online social networks, mobile computing, graph theory, web information retrieval, and evolutionary computation, with funding from national agencies and various university programs and awards. He was the recipient of the President's Award for Distinguished Teaching in 2018.

Terrence S. Tricco received the B.Sc. (Hons.) degree in computer science and physics and the M.Sc. degree in computational science from the Memorial University of Newfoundland, St. John's, NL, Canada, and the Ph.D. degree from the School of Mathematical Sciences, Monash University, Melbourne, VIC, Australia. He has been a Research Fellow with the School of Physics, University of Exeter, Exeter, U.K., CITA Postdoctoral Fellow with the Canadian Institute for Theoretical Astrophysics, University of Toronto, Toronto, ON, Canada, and the Lead Data Scientist for Verafin, St. John's, a Nasdaq company specialized in fighting financial crime. He is currently an Assistant Professor with the Department of Computer Science, Memorial University of Newfoundland.

Cheng Li (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees from the Harbin Institute of Technology, Harbin, China, in 1992 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from the Memorial University of Newfoundland, St. John's, NL, Canada, in 2004. He is currently a Full Professor and the Director of the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada. His research interests include wireless communications and networking, communications signal processing, underwater communication and networking, and mobile ad-hoc and wireless sensor networks. He is an IEEE Communications Society Distinguished Lecturer for the 2021–2022 term. He is a Registered Professional Engineer (P.Eng.) in Canada, and a Member of the IEEE Communication Society, IEEE Computer Society, IEEE Vehicular Technology Society, and IEEE Ocean Engineering Society. He was the recipient of the Best Paper Award in IEEE ICC'2023, GLOBECOM'2017 and ICC'2010. He was the General Co-Chair for the ICNC'23, WINCOM'19, and AICON'19, and TPC Co-Chair for the ICNC'20, MSWiM'14, WiMob'11, and QBSC'10. He was the Co-Chair for various technical symposia/tracks of many international conferences, including IEEE GLOBECOM, IEEE International Conference on Communications, and IEEE Wireless Communications and Networking Conference. He is an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE INTERNET OF THINGS JOURNAL, and *IEEE Network Magazine*.

Ting Hu received the Ph.D. degree in computer science from Memorial University in St. John's, St. John's, NL, Canada, and completed her Postdoctoral training in bioinformatics with Dartmouth College, Hanover, NH, USA. She is currently an Associate Professor with the School of Computing, Queen's University in Kingston, Kingston, ON, Canada. Her research interests include explainable AI, evolutionary computing, and the ethical applications of machine learning in biomedicine. She is an Area Editor of the journal *Genetic Programming and Evolvable Machines*, and an Associate Editor for the Journal *Neurocomputing*. She has served on multiple occasions as the program co-chair for prominent evolutionary computing conferences such as, the ACM Genetic and Evolutionary Computation Conference (GECCO) and European Conference on Genetic Programming (EuroGP).