AmGNN: A Framework for Adaptive Processing of Inter-layer Information in Multi-layer Graph

Huaisheng Zhu, Zongyu Wu, Tianxiang Zhao, and Suhang Wang

The Pennsylvania State University, University Park PA 16802, USA
{hvz5312,zongyuwu,tkz5084,szw494}@psu.edu

Abstract. Graphs play a vital role in various applications. Graph Neural Networks (GNNs) excel at capturing topology information by using a message-passing mechanism to enrich node representations with local neighborhood information. Despite their success in modeling single-layer graphs, real-world scenarios often involve multi-layer graphs where nodes can have multiple edges or relationships represented as different layers. Existing methods of multi-layer graph learning struggle to efficiently process inter-layer information, as they mainly focus on preserving similar layers or shared invariant information, which may not be suitable for all situations. We propose a novel framework called Adaptive Multi-layer Graph Neural Networks (AmGNN) to address this challenge. AmGNN learns shared invariant information for nodes that need it and selectively preserves relevant layers' information for nodes not requiring shared invariance. We introduce multi-layer graph contrastive learning to efficiently capture invariant information and learn weights for adaptive processing. Our experiments on real-world multi-layer graphs validate the effectiveness of AmGNN in node classification tasks.

Keywords: Multi-layer Graph · Node Classification · Graph Neural Networks.

1 Introduction

Graphs are pervasive in the real world, such as social networks [24], recommendation systems [2], and knowledge graphs [21]. To capture the topology information in graphs, Graph Neural Networks (GNNs) typically follow the message-passing mechanism, which aggregates the neighborhood representation of a node to enrich the node's representation. This process enriches node representations and preserves both node feature characteristics and topological structures, benefiting various tasks like node classification [14], link prediction [36] and clustering [29].

Despite the great success of GNNs in modeling graphs, the majority of existing works concentrate on single-layer graphs, where only one edge/relationship exists between any pair of nodes. However, in the real world, a pair of nodes can have multiple edges or relationships, forming what is known as a *multi-layer graph*. For example, Figure 1 shows a 3-layer university's social network with three different types of relations: club interactions (Layer 1), pre-university

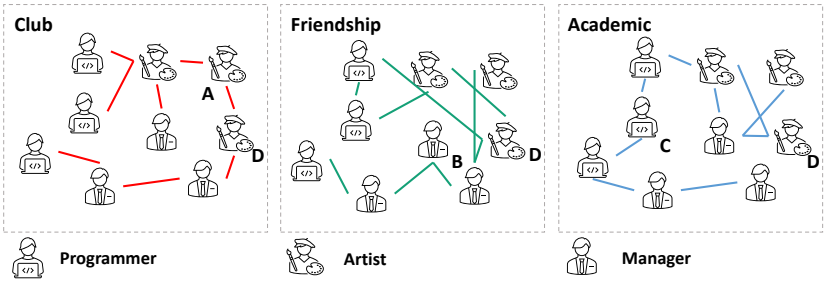

Fig. 1: A motivation example of adaptively processing inter-layer information for multi-layer graph learning. Some nodes require shared invariant information, while others do not.

friendships (Layer 2), and same-major student relations (Layer 3). Simply adopting GNNs designed for single-layer graphs can't achieve good performance on multi-layer graphs. This is because of limited structural information within a single layer and empirical evidence indicating the inadequate performance of single-layer GNNs [18]. Hence, several efforts have been made to model the relations across different layers in multi-layer graphs by aggregating both intra-layer and inter-layer information [18,32]. Generally, they adopt existing GNN models for single-layer graphs to model intra-layer information. For inter-layer aggregation, they aim to preserve shared invariant information (certain common patterns existing in different layers) across layers by relying on correlations between different layers. They typically retain the most similar layers' information to learn representations.

In real-world applications, shared invariant information based on similarity across layers might not always be meaningful for label information, as some nodes' labels could be relevant to specific layers' information. Note that shared invariant information could encompass several layers with higher correlations or might involve most of the similar local neighbors across different layers. Consider the scenario depicted in Figure 1, nodes correspond to students within the university's social network, while edges symbolize interactions occurring in distinct social environments. The labels pertain to the occupations of these students. It shows that individuals' occupations after graduation might be influenced by specific relations or shared patterns across diverse relations. For example, student A's engagement in the Art club could lead to connections with students aspiring to become artists. If we use club relations for predictions, their label might be accurately predicted based on their friends' information. However, the preservation of the most similar or invariant information across layers, like the neighbor patterns of student A in both friendship and academic relations (where the neighbors are labeled as managers), could lead to inaccurate predictions. Similarly, student B is more inclined to prioritize the preservation of friendship relations and student C is more inclined to prioritize the preservation of academic relations. In contrast, certain individuals, like student D, have occupations that are influenced by shared invariant information from three relations, exhibiting

similarities in their local neighborhoods and having most of their friends across different relations belonging to the artist occupation. Assigning higher weights to just one layer could introduce noise, as half of D's neighbors belong to different classes compared to D. This underscores the importance of identifying nodes that need to preserve similar layers' information. Hence, a crucial step is to determine which nodes require shared invariant information across layers and which nodes specifically need information from certain layers. Once this differentiation is made, an adaptive selection process is employed to incorporate important information from different layers for nodes that may benefit from specific layers' information for accurate label prediction. Though promising, the work on adaptively learning layers' information for each node to predict labels of nodes in multilayer graphs is limited.

Therefore, in this paper, we explore a novel problem of node classification in multi-layer graphs, focusing on recognizing nodes that need to preserve shared invariant information while selectively leveraging information from other layers. In essence, there are two main challenges: (**i**) How to determine the nodes requiring invariant information and preserve shared invariant information for node classification?; (**ii**) How to assist nodes, not in need of invariant information, in selecting relevant layer information for node classification? To resolve these challenges, we introduce a novel framework called Adaptive Multi-layer Graph Neural Networks (AmGNN). Firstly, to learn the weights determining whether to preserve invariant information, AmGNN assigns higher weights to nodes with consistent predicted results from pretrained GNN models on single layer relations. By doing so, preserving shared information for these nodes is more likely to yield correct prediction results. Moreover, to efficiently capture invariant information, we deviate from previous approaches that rely on layers' similarity. Instead, we introduce a multi-layer graph contrastive learning method that allows us to assign different weights to the contrastive learning loss for each node, providing control over whether nodes preserve invariant information. Additionally, for nodes without consistent prediction results, we encourage them to preserve the information from specific layers where the pre-trained GNN models yield high-confidence predictions. Our main **contributions** are: (i) We explore a novel problem of adaptive processing for inter-layer information for multi-layer graphs; (ii) We introduce AmGNN, which learns shared invariant information exclusively for nodes that require it and adaptively preserves the relevant layers' information for nodes that may not need shared invariant information; and (iii) Experiments on real-world multi-layer graphs demonstrate the effectiveness of the proposed framework AmGNN.

2 Related Work

Graph Neural Networks. GNNs are popular for node representation learning on graph-structured data and can be categorized into two types: spectral-based [14,28,9,33,8] and spatial-based [30,6,5,34]. Spectral-based GNNs use graph signal processing and apply convolutional operations to graph data in the spec-

tral domain. GCN [14], which uses a first-order approximation, is an example of a spectral-based GNN. Spatial-based GNNs aggregate information from neighboring nodes to update the representation of a given node. For example, Graph Attention Network (GAT) [30] utilizes attention mechanisms to update node representations using different weights.

Graph Contrastive Learning. Contrastive methods have been widely used in graph self-supervised learning [15,6,7]. Graph AutoEncoders [15], for instance, learns node embeddings unsupervisedly by reconstructing the adjacency matrix, while GMI [23] maximizes the mutual information of both features and edges between inputs and outputs. Inspired by the success of contrastive learning, recent work has explored creating positive pairs using reliable graph information instead of data augmentations. For example, SUGRL [20] utilizes two encoder models, one based on GCN and the other an MLP, to generate two sets of node embeddings from different sources. The positive pair for each node is then constructed using its GCN output and MLP output. Similarly, AFGRL [16] and AF-GCL [31] consider nodes in the target node's multi-hop neighborhood as candidate positive examples and employ well-designed similarity measures to select the most similar nodes as positive examples.

GNNs for Multilayer Graphs Multi-layer graph, also known as a multiplex, multi-view, or multi-dimensional graph, takes into account multiple relationships among nodes [17,1,25,18]. Various approaches have been proposed to handle the complexity of multi-layer graphs. Some methods, such as MVE [25] and HAN [32], leverage attention mechanisms to effectively combine embeddings from different views. mGCN [18] addresses interactions within and across views to improve node classification. Node embedding techniques have been explored for node clustering and classification tasks in multi-layer graphs [3,26]. For instance, VANE [3] employs adversarial training to enhance node representation learning comprehensiveness and robustness. Contrastive learning has also found application in learning expressive representations for multi-layer graphs [12,11]. For instance, HDMI [12] learns network embeddings for multi-layer networks by incorporating high-order mutual information and a fusion module based on an attention mechanism. Also, SSDCM [19] maximizes mutual information between local and contextualized global graph summaries using an InfoMax learning strategy, facilitating effective joint modeling of nodes and clusters while utilizing cross-layer links to regularize embeddings across different layers.

Our work is inherently different from existing works: (i) Existing works on multi-layer GNNs concentrate on preserving layers' invariant information across layers; while our study focuses on a novel problem of efficiently processing inter-layer information for multi-layer graphs; (ii) We propose a novel framework that identifies which nodes require shared invariant information and which nodes need specific information from individual layers. Additionally, our framework guides the inter-layer aggregation process adaptively, allowing for selective and effective information propagation across different layers.

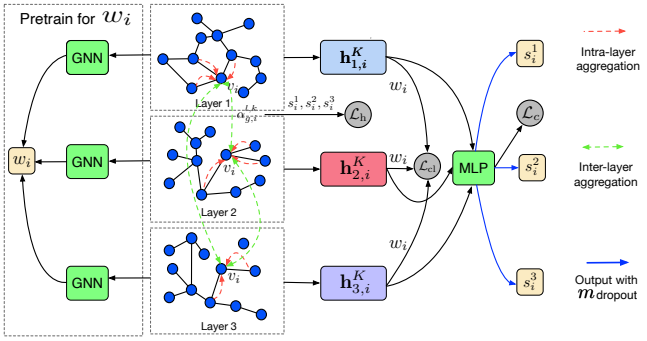

Fig. 2: An overview of the proposed AmGNN. We use a multi-layer graph with $L = 3$ layers as an example.

3 Problem Definition and Notations

We use $\mathcal{G} = \{\mathcal{V}, \mathcal{E}_1, \ldots, \mathcal{E}_L\}$ to denote an L-layer attributed graph, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes, $\mathcal{E}_l \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges in the l-th layer, and $\mathbf{X}$ is the node attribute matrix with $\mathbb{X}[j,:] \in \mathbf{R}^{1\times d}$ being node attribute vector for node v_j. $\mathbf{A}^l$ is the adjacency matrix for the l-th layer. $A^l_{i,j} = 1$ if nodes v_i and v_j are connected in layer l, otherwise $A^l_{i,j} = 0$. Specifically, 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 a multi-layer graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}_1, \ldots, \mathcal{E}_L\}$ *and the partial labels* $\mathcal{Y}_L$*, we aim to learn a node classifier* $Q_\theta(\mathcal{V}, \mathbf{A}^1, ..., \mathbf{A}^L, \mathbf{X}) \to \mathcal{Y}$.

4 Methodology

In this section, the proposed framework is designed for adaptively learning different layers' information of multi-layer graphs. An illustration of the proposed framework is shown in Figure 2. Capturing inter-layer information is crucial for multi-layer graph neural networks to effectively handle node classification tasks. To achieve this, we propose a multi-layer Graph Neural Network framework that incorporates intra-layer aggregation, inter-layer aggregation, and multi-layer graph contrastive learning to model and preserve shared invariant information. However, as depicted in Figure 1, not all nodes necessarily require invariant information; some nodes may benefit from it, while others may require specific layers' information. To address this issue, we introduce node-specific weights for multi-layer graph contrastive learning, enabling us to adaptively control the importance of invariant information for each node. Moreover, AmGNN guides the inter-layer aggregation process, allowing nodes that need specific information to learn relevant layers' information more effectively. Next, we introduce the details.

4.1 Multi-layer Graph Neural Network

To enhance representation learning for multi-layer graphs by aggregating both inter-layer and intra-layer information, we introduce the Multi-layer GNN. Each layer in the graph may possess distinct structural characteristics that can be beneficial for downstream tasks. The design of this module is inspired by mGCN [18]. The first step of Multi-layer GNNs is to model intra-layer information, conventional message passing for single-layer graphs is used to model the intra-layer relations as:

$$\mathbf{T}_l^k = \mathbf{H}^{k-1}\mathbf{W}_l^k, \quad \mathbf{F}_l^k = \sigma\left(Conv(\mathbf{T}_l^k)\right), \quad \mathbf{E}_l^k = Conv_{lin}(\mathbf{T}_l^k), \tag{1}$$

where l is the layer of graphs and k is the layer of the model. $Conv$ means the convolution operation in GNN and $Conv_{lin}$ denotes the linear operation in GNN to make sure $\mathbf{F}_l^k$ and $\mathbf{E}_l^k$ have the same dimension. $\mathbf{W}_l^k$ is the learnable weight matrix utilized to transform the representation of the fused representation to the layer l and σ means activation function. $\mathbf{H}^{k-1}$ is the learned representation matrix from the last layer of the model and $\mathbf{H}^0 = \mathbf{X}$. However, the structure information within a single layer is limited. Thus, independently learning and fusing information from different layers may not adequately capture the diverse layers' information for each node. To tackle this challenge, we propose to enhance the message-passing process by aggregating inter-layer information. Specifically, we aggregate information from other layers for each node using different weights, enabling a more effective modeling of diverse information across the layers. Subsequently, inspired by [18], we combine the aggregated intra-layer and inter-layer information to create a comprehensive representation for each node:

$$\mathbf{H}_l^k = \text{Concat}(\mathbf{F}_l^k, \sum_{g=1, g\neq l}^{L} \boldsymbol{G}_g^{l,k}\mathbf{E}_g^k), \quad \boldsymbol{\alpha}^{l,k} = \text{softmax}([\boldsymbol{z}_1^k, ...\boldsymbol{z}_{l-1}^k, \boldsymbol{z}_{l+1}^k, ..., \boldsymbol{z}_L^k]), \tag{2}$$

where $\boldsymbol{z}_g^k = \text{MLP}(\mathbf{E}_g^k, \mathbf{E}_l^k)$ and softmax represents the row-wise softmax operation. $\boldsymbol{\alpha}^{l,k}$ represents weights used to aggregate information from other layers to layer l. $\boldsymbol{G}_g^{l,k}$ is a diagonal matrix $diag(\boldsymbol{\alpha}_g^{l,k})$ where $\boldsymbol{\alpha}_g^{l,k}$ denotes the weights to aggregate information from layer g to l. Concat represents the concatenation operation. Subsequently, in order to capture information from all layers, following [18], we aggregate the representations from different layers to obtain the representation for the next model layer:

$$\mathbf{H}^k = \sigma\left(\mathbf{W}^k \cdot \text{ Concat }_{l=1}^{L}\mathbf{H}_l^k\right), \tag{3}$$

where $(\cdot)$ means matrix multiplication and $\mathbf{W}^k$ is the learnable parameter. Finally, we obtain the predicted label matrix by the final model layer:

$$\hat{\mathbf{Y}} = \text{MLP}(\mathbf{H}^K), \tag{4}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times C}$ is the predicted probability matrix and C is the number of classes. We adopt cross-entropy to train the model for the node classification as:

$$\mathcal{L}_c = -\sum_{v_i \in \mathcal{V}_L} \sum_{c=1}^{C} Y_i^c \log \hat{Y}_i^c, \tag{5}$$

where Y_i^c is the c-th element of the one-hot encoding of v_i's ground-truth label.

4.2 Multi-layer Graph Contrastive Learning

To effectively aggregate information from different layers in a multi-layer Graph Neural Network using learnable weights, it is vital to develop supervised signals that guide the learning process across these layers. The shared invariant information among different layers holds essential characteristics of both the graph's structure and node features [18]. One common way to preserve shared invariant information across layers is to employ contrastive learning methods, i.e., we treat node representations from various layers as positive samples, promoting the preservation of shared and similar information within their representations. Moreover, we incorporate negative samples to further reinforce the preservation of shared invariant information. To achieve this goal, we introduce a multi-layer graph contrastive learning method across different layers for v_i of the model's final K-th layer as:

$$\mathcal{R}_{\mathcal{N}}^{i,K} = \sum_{l=1}^{L} \sum_{g \neq l}^{L} \log \frac{\mathbf{e}^{\cos\left(\mathbf{h}_{l,i}^K, \mathbf{h}_{g,i}^K\right)}}{\mathbf{e}^{\cos\left(\mathbf{h}_{l,i}^K, \mathbf{h}_{g,i}^K\right)} + \sum_{v_j \in \mathcal{N}_n^{g,i}} \mathbf{e}^{\cos\left(\mathbf{h}_{l,i}^K \mathbf{h}_{g,j}^{K^-}\right)}},$$

where $\mathcal{N}_n^{g,i}$ is set of negative on the layer g for v_i, $\mathbf{h}_{g,j}^K$ is v_j's representation on g-th graph and K-the model layer, and $\cos(\cdot,\cdot)$ denotes the cosine similarity function between two vectors. While many graph contrastive learning methods typically employ data augmentation techniques to generate negative samples, such approaches can be computationally and memory-intensive. To mitigate these problems, we opt for a simpler approach by randomly selecting some negative samples for node v_i that do not have links based on the original graph structure. During the training process, these negative samples form a negative sampling set $\mathcal{N}_n^{g,i}$ for the layer g of the node v_i.

4.3 Adaptive Inter-layer Aggregation

Contrastive learning for multi-layer graphs is effective in preserving invariant information across layers. However, not all nodes necessarily require this shared invariant information. To address this, we propose to assign different weights to the multi-layer contrastive learning loss of different nodes, allowing for selective preservation of shared invariant information based on each node's specific needs. To calculate these weights, as shown in Figure 2, we first train L GNN models

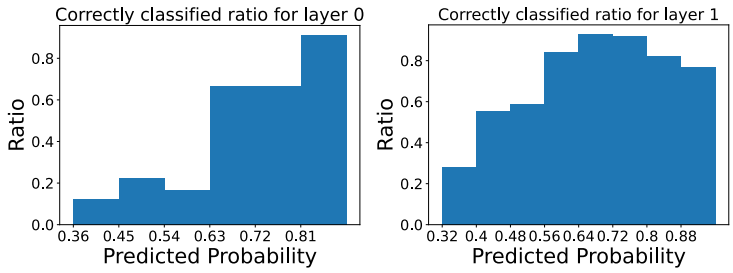

Fig. 3: Predicted probability distribution (correctly classified ratio) for inconsistent nodes on the dataset ACM.

on L graphs respectively to obtain predicted class probabilities for nodes in different layers. Note that this only need to be done once. Let $\mathbf{p}_i^l \in \mathbb{R}^C$ denote the predicted class probability vector of node v_i in layer l, where C denotes the number of classes for node classification. Nodes with the same labels across different layers can be denoted as $\mathbf{p}_i^l = \mathbf{p}_i^{l'}$ for l' ranging from 1 to L and $l \neq l'$. For nodes with consistent predictions, the predicted label can be treated as invariant information across layers. These consistent prediction results are highly confident and have a high probability of accurately representing the ground truth label for node v_i, as models trained separately exhibit similar prediction outcomes. To represent the level of consistent prediction results for nodes in different layers, we first calculate the average prediction results $\mathbf{p}_i$ from pretrained GNN models on different layers:

$$\mathbf{p}_i = \frac{1}{L}\sum\nolimits_{l=1}^{L} \mathbf{p}_i^l, \tag{6}$$

To quantify the level of consistent prediction results, we utilize entropy as a measure, where high entropy indicates low-level consistency and low entropy indicates high-level consistency. Accordingly, we use normalized entropy as the weight to represent the level of nodes with consistent prediction results:

$$w_i = 1 - b_i, \quad b_i = -\frac{\sum_{c=1}^{C} \mathbf{p}_{v_i,c} \log \mathbf{p}_{v_i,c}}{\log C}, \tag{7}$$

where b_i is the normalized entropy of $\mathbf{p}_i$. To ensure that lower or higher weights correspond to higher or lower predicted consistency, and since $b_i \in [0, 1]$, we use $w_i = 1 - b_i$ to denote the weight assigned to nodes for deciding the level of preserving shared invariant information. To help nodes preserve invariant information, we use the multi-layer graph contrastive learning with different weights w_i for different nodes on the model's final Kth layer:

$$\mathcal{L}_{\text{cl}} = -\frac{w_i}{N}\mathcal{R}_N^{i,K}, \tag{8}$$

Training the multi-layer graph contrastive learning may still be time-consuming so we update our model's parameters using this loss every 30 epochs.

Another challenge is about how to preserve useful information for nodes with low-level consistent prediction results because invariant shared information for

these nodes may not be reliable as shown in Figure 1. To address this challenge, we propose to propagate trustworthy information across layers and mitigate the influence of uncertain information. Firstly, we denote the set of nodes which don't have consistent prediction results from separately trained GNN models on different layers as $\mathcal{N}_h$. And we call these nodes inconsistent nodes. We show the empirical results in Figure 3. Specifically, we visualize the predicted probability distribution of nodes' ground-truth classes in relation to the ratio of correctly classified nodes. This visualization is obtained from GNN models trained independently on individual layers. The predicted probability of node v_i' ground-truth classes can be denoted as p^l_{i,c_g}, where c_g is the ground-truth label of the node v_i. Figure 3 shows how the ratio of correctly classified nodes varies across different intervals of predicted probability. From the figure, we observe that predicted results from pretrained GNN models on layers with high-confidence probabilities (higher predicted probabilities for the ground-truth label) hold a high correctly classified ratio. Based on these observations, we will encourage each layer's representation to aggregate layers with highly confident prediction results for nodes $v_i \in \mathcal{N}_h$.

To accurately measure the confidence level of predicted results, relying solely on the value of predicted probability may not provide a comprehensive evaluation. One commonly used approach to assess confidence is by employing dropout techniques [4,13]. Dropout introduces randomness by deactivating neurons during training, creating diverse subnetworks with unique active neurons that lead to prediction variations. When a model is uncertain, distinct subnetworks yield significantly different outputs for the same input, while confident models produce more consistent outputs across dropout instances. Specifically, we start by obtaining different prediction results by applying dropout several times to the model for the layer l. This yields a set of representation vectors $\{\mathbf{H}^K_{1,l}, \ldots, \mathbf{H}^K_{m,l}\}$ from the model. Subsequently, we input these representation vectors into the prediction layer (MLP layer) in Eq. (4) and obtain the results $\{\mathbf{y}^l_1, \ldots, \mathbf{y}^l_m\}$.

To quantify the confidence level of nodes, we calculate the level of consistency among the different prediction results. This consistency measure helps us assess the stability and reliability of the model's predictions, providing valuable insights into the uncertainty estimation of the model. To represent the consistency level of prediction results, we first obtain the average prediction results by generating m results using dropout:

$$\overline{\mathbf{y}}^l = \frac{1}{m}\sum\nolimits_{a=1}^{m} \mathbf{y}^l_a, \tag{9}$$

Furthermore, the entropy measure, as discussed in Eq. (7), can also be utilized to assess the uncertainty level of predicted results obtained from different dropout ratios. This allows us to determine the level of trustable information across layers. The confidence score for the hard classified node v_i is:

$$s^l_i = 1 - \frac{\sum_{c=1}^{C} \overline{y}^l_{i,c} \log \overline{y}^l_{i,c}}{\log C}, \tag{10}$$

Table 1: Statistics of the Datasets

Datasets	Num of Nodes	Edge Types	Num of Edges per Layer	Feature Dimension	Num of Classes
ACM	3025	PSP PAP	2210761 29281	1830	3
IMDB	3550	MAM MDM	66428 13788	1007	3
DBLP	7907	PAP PPP PATAP	144783 90145 57137515	2000	4
Amazon	7621	IVI IBI IOI	266237 1104257 16305	2000	4

where the value of s_i^l can serve as an indicator of which layers' information is considered trustable. To reduce computational overhead, we update s_i^l every 10 epochs.

With s_i^l, we can now encourage the inter-layer aggregation to effectively aggregate trustable information. This is achieved with the following loss function

$$\mathcal{L}_h = -\frac{1}{L\,|\mathcal{N}_h|} \sum_{v_i \in \mathcal{N}_h} \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{g=1, g\neq l}^{L} s_i^g \log \alpha_{g,i}^{l,k}, \tag{11}$$

where $\alpha_{g,i}^{l,k}$ is the weight to control the inter-layer aggregation for the node v_i. Note that the above loss function works for $L \geq 3$ but does not work for $L = 2$. This is because $\alpha_{g,i}^{l,k}$ obtained from Eq.(2) is 1 for a two-layer graph, making $\mathcal{L}_h$ being 0.

4.4 Final Objective Function of AmGNN

Putting everything together, the final objective function of AmGNN is given as:

$$\min_{\Theta} \mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_{\text{cl}} + \gamma \mathcal{L}_h, \tag{12}$$

where θ represents the learnable parameters for our model, λ and γ are hyperparameters to control the loss for multi-layer graph contrastive learning and adaptive inter-layer aggregation.

5 Experiments

In this section, we conduct experiments on real-world multi-layer graphs to demonstrate the effectiveness of AmGNN. In particular, we aim to answer the following research questions: (i) **RQ1** Can AmGNN provide accurate node classification for multi-layer graphs? (ii) **RQ2** What are the contributions of each component for AmGNN? (iii) **RQ3** How reliable is confidence score?

5.1 Datasets

Following previous work [22,12], we conduct experiments on four publicly available real-world multi-layer graphs. The statistics of these datasets are summarized in Table 1 and their details are given below: (i) **ACM** [32]: There are two kinds of edges: paper-author-paper (PAP) and paper-subject-paper (PSP). Each paper has a node feature extracted from its abstract. papers have one of the following labels: Database, Wireless Communication, and Data Mining; (ii) **IMDB**[1]: There are two edge types: movie-actor-movie (MAM) and movie-director-movie (MDM). The movie feature is the bag-of-words feature of movie plots. Each movie belongs to one of the following classes: Action, Comedy, Drama; (iii) **DBLP** [27]: It contains three types of edges: paper-paper-paper (PPP), paper-author-paper (PAP), and paper-author-term-author-paper (PATAP). The bag-of-words feature of paper's abstract represents each paper. Each node has one of the following labels: Data Mining, Artificial Intelligence, Computer Vision, and Natural Language Processing; (iv) **Amazon** [10]: It is sourced from Amazon.com and represents a three-layer graph based on three relations: (1) item-viewed-item (IVI): two items are viewed by the same customer; (2)item-bought-item (IBI): two items are bought by the same customer; (3) item-together-item (IOI): two items are co-bought by a customer. Each node is classified into one of the following classes: Beauty, Automotive, Patio Lawn and Garden, and Baby. Both DBLP and Amazon are three-layer graphs while ACM and IMDB are two-layer graphs. Note that though our $\mathcal{L}_h$ only works for graphs of three or more layers, we also include two-layer graphs to show that our method can also achieve good results on two-layer graphs.

5.2 Experimental Setup

Baselines. We compare AmGNN with representative and state-of-the-art methods for node classification, which include:

- **GCN** [14]: GCN is one of the most popular spectral GNN models based on graph Laplacian, which has shown great performance for node classification.
- **GAT** [30]: GAT adopts an attention mechanism for enhanced neighborhood aggregation during message-passing.
- **GIN** [35]: Graph Isomorphism Networks (GIN) employ multi-layer perceptrons to process the aggregated information from neighbors at each layer, enabling the model to learn more potent and expressive node representations.
- **GCN/GAT/GIN-C**: As GCN/GAT/GIN can only deal with single layer graph, for GCN/GAT/GIN-C, we combine edge information from all layers to construct one adjacency matrix. Then, we adopt them on the combined adjacency matrix for node classification.
- **mGCN** [18]: mGCN utilizes GCN to extract node embedding for each layer of the multi-layer graphs and then combine them via the attention mechanism.

[1] https://www.imdb.com/

Table 2: Node Classification Performance. The best and second-best performances are marked with boldface and underlined.

Dataset	ACM		IMDB		DBLP		Amazon	
Metric	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
GCN	91.08±0.39	91.14±0.38	66.95±0.28	66.97±0.32	82.82±0.63	83.64±0.68	75.82±0.15	75.22±0.04
GAT	90.53±0.17	90.51±0.16	67.30±0.26	67.20±0.24	81.95±0.26	82.74±0.29	75.65±0.78	75.10±0.85
GIN	89.54±0.04	89.54±0.04	62.44±0.85	62.32±0.92	81.92±0.80	82.75±0.79	74.89±0.23	74.26±0.39
GCN-C	67.31±0.44	63.98±0.98	62.42±0.68	62.62±0.70	77.13±0.51	77.20±0.47	72.09±0.24	67.24±0.54
GAT-C	68.71±0.29	62.79±0.75	62.59±0.47	61.38±0.93	77.42±0.63	77.60±0.54	71.49±0.31	67.82±0.65
GIN-C	69.52±0.31	63.42±0.49	62.19±0.59	62.77±0.93	76.98±0.43	77.11±0.38	66.39±0.48	62.60±0.26
HAN	92.87±0.26	92.94±0.35	69.61±0.32	69.05±0.54	84.07±0.43	84.12±0.31	77.57±0.46	77.28±0.27
mGCN	92.63±0.58	91.70±0.34	67.84±0.47	67.93±0.48	83.91±0.39	83.27±0.46	85.98±0.87	85.59±0.97
SSDCM	92.99±0.41	92.32±0.20	68.65±0.22	67.55±0.27	83.75±0.28	83.76±0.37	84.18±0.26	86.11±0.29
HDMI	92.72±0.10	92.78±0.10	67.44±0.56	67.71±0.54	**84.08±0.11**	**84.83±0.10**	86.49±0.56	86.78±0.46
X-GOAL	93.56±0.07	93.38±0.06	69.48±0.32	69.71±0.33	83.54±0.13	84.57±0.14	89.04±0.12	89.94±0.13
AmGNN	93.58±0.06	93.61±0.05	69.93±0.99	69.98±1.05	83.02±0.20	83.70±0.25	89.33±0.67	89.18±0.68
Ours+x-GOAL	**94.87±0.45**	**94.91±0.43**	**70.43±0.31**	**70.51±0.35**	83.33±0.14	84.08±0.20	**90.36±0.43**	**90.19±0.39**

– **HAN** [32]: The model learns node embeddings specific to different metapaths from different relations. It then employs the attention mechanism to combine these embeddings into a single vector representation for each node.
– **SSDCM** [19]: It uses an InfoMax to maximize mutual information between local and contextualized global graph summaries. It also leverages cross-layer links to regularize the embeddings across different layers of the graph.
– **HDMI** [12]: HDMI adopts a new contrastive learning loss by capturing high-order information across different layers.
– **X-GOAL** [11]: It includes a GOAL framework, which learns node embeddings for each graph layer, and an alignment regularization to model and propagate information across different layers. After obtaining node embeddings for both HDMI and X-GOAL, we employ MLPs to perform node classification.

Configurations. All experiments are conducted on a machine with Nvidia GPU (NVIDIA RTX A6000, 48 GB memory) and a machine with Nvidia GPU (NVIDIA RTX A100, 80 GB memory). The learning rate is initialized to 0.001. Besides, all models are trained until convergence, with the maximum training epoch being 1000. The implementations of all baselines are based on Pytorch Geometric or their source code. Train/eval/test splits are set to 3/1/6. The hyperparameters of all methods are tuned on the validation set.
Evaluation Metrics. Following existing works on node classification [14,11], we adopt Accuracy and Macro F1-score as the evaluation metrics.

5.3 Node Classification Performance

In this subsection, we compare the performance of the proposed method with baselines for node classification on the multi-layer graphs, which aims to answer **RQ1**. Each experiment is conducted 5 times. Means and standard deviations are reported in Table 2. From the table, we observe: (**i**) Compared with GCN/GAT/GIN, the performance of GCN/GAT/GIN-C becomes worse on most datasets. It demonstrates that simply combining information from other layers

can have a negative effect on the node classification task. Our proposed method can further outperform both GCN/GAT/GIN-C and GCN/GAT/GIN, which verifies the effectiveness of our method to greatly learn structure information in different layers for multi-layer graphs; (**ii**) Our model can also consistently outperform multi-layer GNN models (HAN, mGCN and SSDCM) on most datasets. This verifies our motivation that it's necessary to adaptively preserve shared invariant information and select layers information for each node to facilitate the node classification task; (**iii**) Both X-GOAL and HDMI are state-of-the-art baselines, which adopt contrastive learning on multi-layer graphs. They show greater performance compared with other models. Our model can further outperform them on most datasets. This is because our method can effectively capture inter-layer relations. The potential reason for our method's suboptimal performance on the DBLP dataset is that using the concatenation operation in Eq. 2 can cause an out-of-memory (OOM) issue. Therefore, we replace the concatenation with the addition operation. Furthermore, using embeddings from X-GOAL can further improve the performance.

5.4 Ablation Study

To answer **RQ2**, in this subsection, we conduct an ablation study to evaluate the contribution of each component in AmGNN. Specifically, we consider the following ablations: (**i**) w/o adp, which is a variant by removing the main component that controls the inter-layer aggregation process, i.e, we set γ as 0 for Eq. (12) . (**ii**) w/o CL, which denotes the variant by removing the key component of multi-layer graph contrastive learning, i.e., we set λ as 0 for Eq. (12). (**iii**) w/o both, which means that we remove these two components. The results are shown in Table 3. We only report the results on Amazon because we can have similar observations on DBLP. All experiments are conducted five times and the means and standard deviations are reported. We can observe that only preserving the shared invariant information across layers (without adp) with multi-layer graph contrastive learning loss can still learn expressive representation for node classification. This observation highlights the effectiveness of our AmGNN in identifying nodes that require the preservation of shared invariant information. Additionally, we have observed that solely adaptively controlling the inter-layer aggregation process (without contrastive learning) can improve the performance of our designed multi-layer graph neural networks. This observation confirms that AmGNN is capable of assisting nodes in selecting trustworthy layers' information for inter-layer aggregation. While we remove these two key components, the performance is worse compared with removing only one component. Our experimental findings confirm the validity of our approach to extract invariant information across layers and selectively aggregate inter-layer information, leading to improved node classification performance. Importantly, the full

Table 3: Ablation Study on Amazon.

Dataset	Amazon	
Metric	Accuracy	F1-score
w/o adp	88.93±0.70	88.77±0.70
w/o CL	88.97±0.68	88.68±0.72
w/o both	88.75±0.83	88.43±0.88
AmGNN	**89.33±0.67**	**89.18±0.68**

model (last row) achieves the highest performance, highlighting the complementary nature of the different components in AmGNN.

5.5 Analysis of Confidence Score

To answer **RQ3**, we conduct the visualization experiment in this subsection. Our objective is to investigate the ability of the weights s_i^l in Eq. (10) to identify nodes that require which layers information. For GNNs that focus on aggregating local neighbors' information, nodes that share similar labels with neighbors may have a higher probability to be correctly classified [37]. Hence, nodes opting for layers with higher label matching ratios concerning neighbors in that layer can be deemed as reliable layers, facilitating node classification performance.

To quantify this, we first compute the ratio of neighbors that share the same class as the central nodes. Specifically, we denote the neighbors set of the node v_i as $\mathcal{N}_i^l$ and a set of neighbors having similar labels as v_i as $\mathcal{S}_i^l$. For $v_j \in \mathcal{S}_i^l$, we have $\mathbf{Y}_j = \mathbf{Y}_i$ and $v_j \in \mathcal{S}_i^l$. The label matching ratio for v_i in the layer l is $r_i^l = |\mathcal{S}_i^l|/|\mathcal{N}_i^l|$. And we denote the vector $r_i \in \mathbb{R}^L$, where each dimension is r_i^l. Similarly, we employ the vector $\mathbf{s}_i \in \mathbb{R}^L$ to represent the same concept, where each dimension is denoted as s_i^l. To verify whether $\mathbf{s}_i$ captures relevant layers information to improve the performance of node classification, we calculate the cosine similarity between this local label matching ratio $\mathbf{r}_i$ and $\mathbf{s}_i$. Finally, we visualize the distribution of these cosine similarity values on them, and the results are shown in Figure 4. We observe that the majority of cosine similarity values exceed 0.5. The substantial similarity between $\mathbf{s}_i$ and $\mathbf{r}_i$ shows the effectiveness of the confidence score.

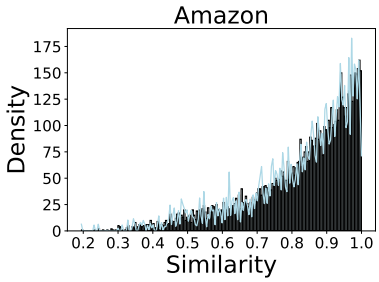

Fig. 4: Visualization Experiments for AmGNN.

6 Conclusion and Future Work

In conclusion, this paper addresses the problem of node classification in multi-layer graphs by selectively leveraging information from different layers. we propose a novel framework called Adaptive Multi-layer Graph Neural Networks (AmGNN). AmGNN efficiently captures invariant information by introducing a multi-layer graph contrastive learning method that assigns different weights to the contrastive learning loss for each node. We also develop a strategy to determine whether nodes require shared invariant information based on their consistency in predicted results from pretrained GNN models on single layer relations. Additionally, we encourage nodes without consistent prediction results to preserve information from layers with high-confidence predictions. Empirical results on real-world multi-layer graphs demonstrate the effectiveness of AmGNN in node classification tasks.

Several interesting directions need further investigation. First, one limitation is that the concatenation operation in Eq. 3 requires large GPU memory and will cause OOM problems on large-scale graphs. One future direction is how to reduce the GPU memory that our framework needs. Second, for simplicity, we randomly select negative samples for a given node from nodes that are not linked in the original graph. More complicated methods could be used to find negative nodes.

Acknowledgments. This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STCIN00001-05-00. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Cen, Y., Zou, X., Zhang, J., Yang, H., Zhou, J., Tang, J.: Representation learning for attributed multiplex heterogeneous network. In: Proceedings of SIGKDD (2019)
2. Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In: Proceedings of WWW. pp. 417–426 (2019)
3. Fu, D., Xu, Z., Li, B., Tong, H., He, J.: A view-adversarial framework for multi-view network embedding. In: Proceedings of CIKM (2020)
4. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: Proceedings of ICML. pp. 1050–1059 (2016)
5. Gao, H., Wang, Z., Ji, S.: Large-scale learnable graph convolutional networks. In: Proceedings of SIGKDD. pp. 1416–1424 (2018)
6. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of NeurIPS (2017)
7. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584 (2017)
8. He, M., Wei, Z., Wen, J.R.: Convolutional neural networks on graphs with chebyshev approximation, revisited. arXiv preprint arXiv:2202.03580 (2022)
9. He, M., Wei, Z., Xu, H., et al.: Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In: Proceedings of NeurIPS (2021)
10. He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of WWW (2016)
11. Jing, B., Feng, S., Xiang, Y., Chen, X., Chen, Y., Tong, H.: X-goal: Multiplex heterogeneous graph prototypical contrastive learning. In: CIKM. pp. 894–904 (2022)
12. Jing, B., Park, C., Tong, H.: Hdmi: High-order deep multiplex infomax. In: Proceedings of WWW. pp. 2414–2424 (2021)
13. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? In: Proceedings of NeurIPS (2017)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
15. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)

16. Lee, N., Lee, J., Park, C.: Augmentation-free self-supervised learning on graphs. In: Proceedings of AAAI. pp. 7372–7380 (2022)
17. Li, J., Chen, C., Tong, H., Liu, H.: Multi-layered network embedding. In: Proceedings of SDM. pp. 684–692 (2018)
18. Ma, Y., Wang, S., Aggarwal, C.C., Yin, D., Tang, J.: Multi-dimensional graph convolutional networks. In: Proceedings of SDM. pp. 657–665 (2019)
19. Mitra, A., Vijayan, P., Sanasam, R., Goswami, D., Parthasarathy, S., Ravindran, B.: Semi-supervised deep learning for multiplex networks. In: Proceedings of SIGKDD (2021)
20. Mo, Y., Peng, L., Xu, J., Shi, X., Zhu, X.: Simple unsupervised graph representation learning. In: Proceedings of AAAI (2022)
21. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1), 11–33 (2015)
22. Park, C., Kim, D., Han, J., Yu, H.: Unsupervised attributed multiplex network embedding. In: Proceedings of AAAI. pp. 5371–5378 (2020)
23. Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., Huang, J.: Graph representation learning via graphical mutual information maximization. In: Proceedings of WWW. pp. 259–270 (2020)
24. Qu, L., Zhu, H., Zheng, R., Shi, Y., Yin, H.: Imgagn: Imbalanced network embedding via generative adversarial graph networks. In: Proceedings of SIGKDD (2021)
25. Qu, M., Tang, J., Shang, J., Ren, X., Zhang, M., Han, J.: An attention-based collaboration framework for multi-view network representation learning. In: Proceedings of CIKM. pp. 1767–1776 (2017)
26. Sun, Y., Wang, S., Hsieh, T.Y., Tang, X., Honavar, V.: Megan: A generative adversarial network for multi-view network embedding. In: Proceedings of IJCAI (2019)
27. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of SIGKDD. pp. 990–998 (2008)
28. Tang, S., Li, B., Yu, H.: Chebnet: Efficient and stable constructions of deep neural networks with rectified power units using chebyshev approximations. arXiv preprint arXiv:1911.05467 (2019)
29. Tsitsulin, A., Palowitch, J., Perozzi, B., Müller, E.: Graph clustering with graph neural networks. arXiv preprint arXiv:2006.16904 (2020)
30. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
31. Wang, H., Zhang, J., Zhu, Q., Huang, W.: Augmentation-free graph contrastive learning with performance guarantee. arXiv preprint arXiv:2204.04874 (2022)
32. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: Proceedings of WWW (2019)
33. Wang, X., Zhang, M.: How powerful are spectral graph neural networks. In: Proceedings of ICML. pp. 23341–23362 (2022)
34. Xiao, T., Chen, Z., Wang, D., Wang, S.: Learning how to propagate messages in graph neural networks. In: Proceedings of SIGKDD. pp. 1894–1903 (2021)
35. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: Proceedings of ICLR (2018)
36. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: Proceedings of NeurIPS (2018)
37. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Beyond homophily in graph neural networks: Current limitations and effective designs. In: Proceedings of NeurIPS (2020)