VR-GNN: Variational Relation Vector Graph Neural Network for Modeling Homophily and Heterophily

Fengzhao Shi
School of Cyber Security, University of Chinese Academy of Sciences
Ren Li
School of Cyber Security, University of Chinese Academy of Sciences
Yanan Cao
caoyanan@iie.ac.cn
School of Cyber Security, University of Chinese Academy of Sciences
Xixun Lin
School of Cyber Security, University of Chinese Academy of Sciences
Yanmin Shang
School of Cyber Security, University of Chinese Academy of Sciences
Chuan Zhou
Chinese Academy of Sciences
Jia Wu
Macquarie University
Shirui Pan
Griffith University

Research Article

Keywords: Data mining, Graph neural networks, Semi-supervised node classification, Social network

Posted Date: January 11th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3842969/v1

VR-GNN: Variational Relation Vector Graph Neural Network for Modeling Homophily and Heterophily

Fengzhao Shi[1,2†], Ren Li[1,2†], Yanan Cao[1,2*], Xixun Lin[1,2], Yanmin Shang[1,2], Chuan Zhou[3], Jia Wu[4], Shirui Pan[5]

[1*]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China.
[2]Institute of Information Engineering, University of Chinese Academy of Sciences, Beijing, China.
[3]Academy of Mathematics and Systems Science, University of Chinese Academy of Sciences, Beijing, China.
[4]School of Computing, Macquarie University, Sydney, Australia.
[5]School of Information and Communication Technology, Griffith University, Gold Coast, Australia.

*Corresponding author(s). E-mail(s): caoyanan@iie.ac.cn;
Contributing authors: shifengzhao@iie.ac.cn; liren@iie.ac.cn; linxixun@iie.ac.cn; shangyanmin@iie.ac.cn; zhouchuan@amss.ac.cn; jia.wu@mq.edu.au; s.pan@griffith.edu.au;
†These authors contributed equally to this work.

Abstract

Graph Neural Networks (GNNs) have achieved remarkable success in diverse real-world applications. Traditional GNNs are designed based on homophily, which leads to poor performance under heterophily scenarios. Most current solutions deal with heterophily mainly by modeling the heterophily edges as data noises or high-frequency signals, treating all heterophilic edges as being of the same semantic. Consequently, they ignore the rich semantic information of these edges in heterophily graphs. To overcome this critic problem, we propose a novel GNN model based on relation vector translation named as **V**ariational **R**elation Vector **G**raph **N**eural **N**etwork (**VR-GNN**). VR-GNN models relation generation and graph aggregation into an end-to-end model based on a variational inference framework. To be specific, the encoder utilizes the structure, feature and label to generate a fine-grained relation vector for each edge, which aims to infer its implicit semantic information. The decoder incorporates the generated relation vectors into the message-passing framework for deriving better node representations. We conduct extensive experiments on eight real-world datasets with different homophily-heterophily properties to verify model effectiveness. Extensive experimental results show that VR-GNN gains consistent and significant improvements against existing strong GNN methods under heterophily and competitive performance under homophily.

Keywords: Data mining, Graph neural networks, Semi-supervised node classification, Social network

1 Introduction

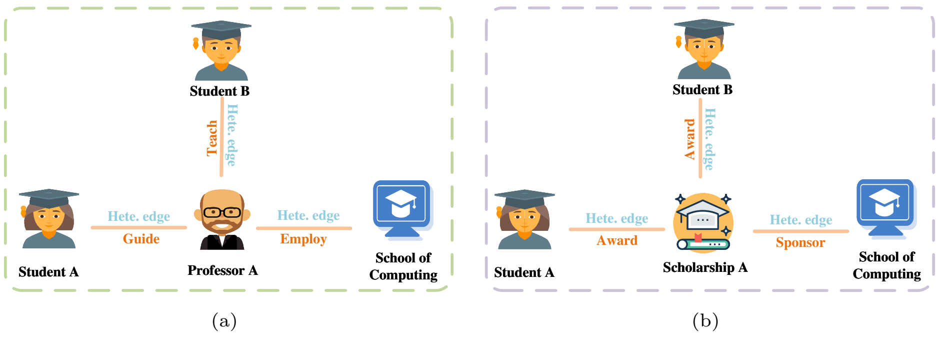


Fig. 1: Modeling the heterophilic edges. The central nodes in the above figures are professor A and scholarship A respectively. The neighbors of two central nodes are the same. The blue represents treating all heterophilic edges as being of the same semantic. The red is the implicit semantic of the edge.

Graph Neural Networks (GNNs) have revealed superior performance on various real-world graph applications, ranging from social networks [1], citation networks [2] to biological networks [3]. Graph Convolutional Network (GCN) [4] and its variants [5, 6] learn node representations by smoothing the features between neighbor nodes. The smoothing operation is suitable for homophilic graphs [7], where connected nodes tend to possess similar features and belong to the same class. However, on heterophilic graphs [8] connected nodes have dissimilar features and different labels, the traditional GCNs suffer from poor performance and even is inferior to Multi-layer Perceptron (MLP) that completely ignores the graph structure [9].

Recently several efforts have been proposed to achieve heterophily-based GNNs. The current approaches are mainly to consider the heterophilic edges as a single semantic to complete the overall modeling, such as treating the heterophilic edge modeling as a specific data noise [10, 11] or high-frequency signal [12–14]. As a result, all these methods ignore the rich semantic representation of the edges (especially the heterophilic edges) in the heterophily graphs, which leads to an unsatisfactory model performance. As shown in figure 1, professor A and scholarship A contain the same neighbors, and the neighbors are heterophilic connected with them. If we regard these edges as the same semantic, then professor A and scholarship A will become more and more similar after GNN aggregation, which is obviously unreasonable. In fact, we can easily distinguish the professor A and scholarship A by considering the semantics of their adjacent edges. For example, student A and professor A should have a "guide" semantic, while scholarship A and student A should have a "award" semantic. Generally, although two nodes have the same neighbors, the semantics of the adjacent edges can be quite different.

To address the above problem, we introduce the concept of relation vector into the message-passing framework for modeling the implicit semantic of each edge, where the message passing between nodes can be described by an addition translation of the relation vector. It is an intuitive idea motivated by recent progress on knowledge graph embedding [15]. To verify the effectiveness of our intuition, we design a preliminary experiment that including three heuristic baselines to show how introducing fine-grained relation vectors can influence the representation in the hetrophily graphs. Concretely, we first use structure, feature and label information of the original graph respectively to divide the edges into fine-grained discrete relation vectors. Then, we change the aggregation strategy of GCN [4] to combine the relation vectors of each edge to generate the node representations. The experimental results on both homophily and heterophily datasets show that an obvious improvement by introducing fine-grained relation vector modeling.

Based on the above empirical analysis, we present **V**ariational **R**elation Vector **G**raph **N**eural **N**etwork (**VR-GNN**), a variational GNN model that innovatively incorporates the fine-grained relation learning into the message-passing process for modeling both homophily and heterophily on graphs. VR-GNN models the relation vector generation and node representation in a unified framework where encoder achieves relation vector generation and decoder achieves node representation, which can transfer the knowledge learned by relation vector into node representations more directly. The encoder of VR-GNN treats relation vectors as latent variables and adopts variational inference to generate it based on the graph structure, feature and label, which combines all the information on the graph to make the relation vector more informative. The decoder incorporates generated relation vectors into the message-passing mechanism, where messages are computed by translating neighbors along connections. Finally, the model takes the output node representation to perform the downstream node classification task. The contributions of our work are summarized here:

- We propose a novel idea that introduces the concept of relation vector into the message-passing framework for modeling the implicit semantic of each edge in heterophily graphs, and provide the empirical analysis for demonstrating the effectiveness of relation vector in learning node representations for heterophily graphs.

- To further fulfill the proposed idea, we present VR-GNN which jointly models the relation vector generation and node representation within a encoder-decoder variational framework.
- We conduct extensive experiments on 8 common homophily and heterophily datasets and the results demonstrate that VR-GNN gains consistent and significant improvements against multiple strong GNN methods under heterophily and competitive performance under homophily.

2 Preliminary

2.1 Problem Definition

A graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of nodes $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ and a set of edges or connections $\mathcal{E}$. The connections of a graph can be described by its adjacency matrix $\mathbf{A} = [a_{ij}] \in \{0, 1\}^{N \times N}$, where $N = |\mathcal{V}|$ is the number of nodes, and $a_{ij} = 1$ means node v_i and v_j has a connection e_{ij} between them. The node feature matrix of a graph can be denoted as $\mathbf{X} \in \mathbb{R}^{N \times F}$, where F is the feature dimensional size of per node. $\mathbf{x}_i \in \mathbb{R}^F$ denotes the i-th row of $\mathbf{X}$ and corresponds to the feature of node v_i. In this paper, we focus on the semi-supervised node classification task, which aims to learn a mapping $f : \mathcal{V} \to \mathcal{C}$, where $\mathcal{C} = \{c_1, c_2, \cdots, c_M\}$ is the label set with M classes, given $\mathbf{A}$, $\mathbf{X}$ and partially labeled nodes $\{(v_1, y_1), (v_2, y_2), \dots\}$ with $y_i \in \mathcal{C}$.

2.2 Message-passing Framework

Most current GNNs apply the message-passing framework [16] to formulate their workflow. Normally the message-passing framework consists of L layers, and in each layer l, it aggregates neighbor and central nodes with the following principle:

$$\mathbf{h}_i^{l+1} = \mathrm{U}(\mathbf{h}_i^l, \mathrm{AGG}(\mathbf{h}_j^l : j \in \mathcal{N}_i)) \tag{1}$$

where $\mathbf{h}_i^l$ denotes the embedding of node v_i in layer l; $\mathcal{N}_i$ is the neighbor set of node v_i; $\mathrm{AGG}(\cdot)$ is the neighbor aggregation function; $\mathrm{U}(\cdot)$ is the updating function of renewing the central node embedding with aggregation information and original node information.

2.3 Homophily Ratio

Here we introduce the concept of homophily ratio [10] to estimate the homophily level of a graph.

Definition 1 (Homophily ratio)**.** The homophily ratio of a node v_i is the proportion of its neighbors belonging to the same class with v_i. The homophily ratio of a graph is the mean of homophily ratios of all its nodes:

$$\mathcal{H} = \sum_{v_i \in \mathcal{V}} \frac{|\{e_{ij} : e_{ij} \in \mathcal{E} \wedge y_i = y_j\}|}{|\mathcal{N}_i|} \in [0,\ 1]$$

Table 1: Performance comparison of GCN+RV model and only GCN. Bold text denotes GCN+RV surpass the original GCN.

Models	**Homophily Dataset**		**Heterophily Dataset**	
	Cora	CiteSeer	Chameleon	Squirrel
GCN	87.14	79.86	59.61	46.78
GCN+RV (structure)	**87.63**	**80.45**	**64.32**	**51.25**
GCN+RV (feature)	**87.54**	**79.93**	**65.79**	**52.34**
GCN+RV (label)	80.32	71.05	**67.33**	**52.98**

A high homophily ratio represents the graph owns a strong homophily property, and a low homophily ratio indicates a weak homophily property or strong heterophily property.

3 Effectiveness Verification of Relation Vector

In this section, we attempt to conduct a preliminary exploration of the modeling of relation vector, to verify whether it is a valuable approach to model homophily and heterophily connections.

The basic idea of relation vector is to refine the rough description of connections in a fine-grained manner. Here we consider a simple refining idea that for each connection (edge), it can be categorized by some common relations according to its characteristics. Then we design three dividing strategies from different aspects as follows:

- Structure aspect: We randomly divide edges into K relations, where K is hyper-parameter. Because there is no guidance for the relation dividing, the subsequent learning of relation vector is based on the graph structure information.
- Feature aspect: For each edge e_{ij}, we categorize it according to the feature difference of its two end points $\|\mathbf{x_i} - \mathbf{x_j}\|$. Specifically, we averagely divide the edges into K ranges with an ascending order of the feature difference. Then we consider that the edges in the same range own the same relation.
- Label aspect: We categorize an edge by the labels of its two end nodes. There is total M^2 relations, and M is the number of node class. Note that here we only use the node labels in training set, and for the unknown node we randomly assign a label to it.

Above three strategies can be regarded as three simple baseline models, which take advantage of different characteristics to refine edges with specific relations. Then each relation will be initialized with a corresponding embedding representation and assigned to the characterized edges. We denote the relation vector on edge e_{ij} as $\mathbf{r}_{ij}$.

To merge the inferred relation vector into the message-passing framework, we consider the classical two-layer Graph Convolutional Network (GCN) model. We simply modify the GCN framework and add the relation vector into neighbor aggregation process:

$$\mathbf{x}_i' = \sigma(\sum_{j \in \mathcal{N}_i} \frac{\mathbf{x}_j}{\sqrt{d_i}\sqrt{d_j}}\mathbf{W}) \;\Rightarrow\; \mathbf{x}_i' = \sigma(\sum_{j \in \mathcal{N}_i} \frac{\varphi(\mathbf{x}_j, \mathbf{r}_{ji})}{\sqrt{d_i}\sqrt{d_j}}\mathbf{W})$$

where $\mathbf{x}_i$ is the embedding of node i; $\mathbf{x}'_i$ is the output embedding of node i after one GCN layer; $\mathcal{N}_i$ is the neighbor set of node i; d_i is the degree of node i; $\mathbf{W}$ and σ are linear and non-linear transformation. Especially, we fuse the relation vector with node embedding using function φ, which is set as $\varphi(\mathbf{x}, \mathbf{r}) = \mathbf{x} + \mathbf{r}$. For simplicity, we only maintain the relation in first GCN layer, but with the layer deepening, the relation vector r will also be propagated and capture the wide-range dependency.

Then we compare the model of GCN+RV (RV represents relation vector.) with only GCN on four frequently used homophily and heterophily datasets including Cora, Citeseer, Chameleon and Squirrel. The results are demonstrated in Table 1. We can see that though the three baselines are very simple, they still bring an evident improvement to the original GCN model.

For the structure aspect baseline, it just categorizes the relation randomly instead of introducing any information to it, which brings an effective performance improvement for GCN on four datasets. It shows that the design itself with relation vectoron edges can bring potential capacity for modeling homophily and heterophily. We also observe that the improvement on heterophily dataset is more evident on homophily dataset. It demonstrates that two nodes with the heterophily connection have a more complicated implicit relationship compared to the one under homophily situation. Such as for label aspect baseline with M classes, if two nodes' label are similar, there are only M situations, while for being dissimilar there are $M^2 - M$ situations (consider directed edges). In generally speaking, the above improvements of three baseline models give a preliminary verification of the effectiveness of relation vector for fine-gained connection modeling. And in the next section, we give a comprehensive framework for modeling the relation vectors.

4 Methodology

4.1 VR-GNN Framework

The core idea of VR-GNN is to introduce relation vectors to describe diverse homophilic and heterophilic connections of the graph within a more effective message passing framework. We treat such a process as an encoder-decoder paradigm, which firstly encodes the connection characteristics into relation vectors and then decodes the relation vectors through GNN to complete the node classification.

In this work, we take Variational Auto-Encoder (VAE), a popular probabilistic technique to encode/decode hidden embedding of the data [17], as our overall framework. Specifically, we treat the relation vector of graph connections as latent variable $\mathbf{z}$, and the node classification as a prediction process guided by $\mathbf{z}$, hence the process of VR-GNN can be formularized as following:

$$p_\theta(\mathbf{Y}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) = \int p_\theta(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})\, p_\theta(\mathbf{Y}|\mathbf{z}, \mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})\, \mathrm{d}\mathbf{z} \tag{2}$$

where $\mathbf{A}$ is adjacency matrix; $\mathbf{X}$ is node feature matrix; $\mathbf{Y} \in \mathbb{R}^{N \times M}$ is label matrix; $\mathbf{Y}_{tr}$ is training label matrix; θ denotes learnable parameters.

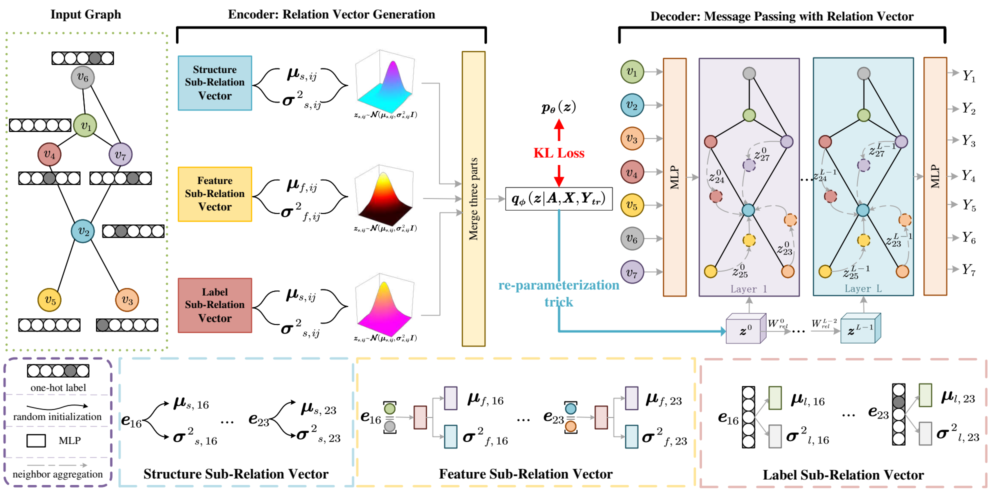

Input Graph
Encoder: Relation Vector Generation
Decoder: Message Passing with Relation Vector
Structure Sub-Relation Vector
Feature Sub-Relation Vector
Label Sub-Relation Vector
Merge three parts
KL Loss
re-parameterization trick
MLP
Layer 1
Layer L
one-hot label
random initialization
MLP
neighbor aggregation
Structure Sub-Relation Vector
Feature Sub-Relation Vector
Label Sub-Relation Vector

Fig. 2: The architecture of VR-GNN. It consists of two components: an encoder to generate relation vectors by combining structure, feature and label information, and a decoder to achieve node classification with generated relation vectors. The encoder generates three types of sub-relation with variational inference and composes them into final relation vectors. The three sub-relations are set as multivariate normal distribution with mutual independence. The decoder utilizes the relation vectors to translate original neighbor features into proper messages and aggregates them for obtaining final node representations.

Since the true posterior $p_\theta(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})$ is intractable, we adopt variational inference [18] to learn it. We introduce a variational distribution $q_\phi(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})$, parameterized by ϕ, to approximate $p_\theta(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})$, and we aim to minimize KL divergence between the two distributions:

$$\min \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) \| p_\theta(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})\right] \tag{3}$$

then following the standard derivation of variational inference, we can get the evidence lower bound (ELBO) learning object:

$$\begin{aligned}\max \mathcal{L}_{(\theta,\phi)} =& - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) || p(\mathbf{z})\right] + \\ & \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})}\left[\log p_\theta(\mathbf{Y}_{tr}|\mathbf{z}, \mathbf{A}, \mathbf{X})\right] \\ =& \mathcal{L}_{en} + \mathcal{L}_{de}\end{aligned} \tag{4}$$

The specific proof of equation 4 is as follows:

Proof 1. Equation 3 can be transformed with following steps:

$$
\begin{aligned}
&\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})\|p_\theta(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})\right]\\
=&\,\mathbf{E}_{\mathbf{z}\sim q_\phi}\left[\log q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})-\log p_\theta(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})\right]\\
=&\,\mathbf{E}_{\mathbf{z}\sim q_\phi}\left[\log q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})-\log\frac{p_\theta(\mathbf{z},\mathbf{Y}_{tr}|\mathbf{A},\mathbf{X})}{p_\theta(\mathbf{Y}_{tr}|\mathbf{A},\mathbf{X})}\right]\\
=&\,\mathbf{E}_{\mathbf{z}\sim q_\phi}\left[\log q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})-\log p_\theta(\mathbf{z},\mathbf{Y}_{tr}|\mathbf{A},\mathbf{X})\right]+\\
&\,\mathbf{E}_{\mathbf{z}\sim q_\phi}[p_\theta(\mathbf{Y}_{tr}|\mathbf{A},\mathbf{X})]\\
=&-\mathcal{L}_{(\theta,\phi)}+p_\theta(\mathbf{Y}_{tr}|\mathbf{A},\mathbf{X})\geq 0
\end{aligned}
$$

Therefore, maximizing the log-likelihood $p_\theta(\mathbf{Y}_{tr}|\mathbf{A},\mathbf{X})$ is equivalent to maximizing its ELBO:

$$
\begin{aligned}
\mathcal{L}_{(\theta,\phi)}=&-\mathbf{E}_{\mathbf{z}\sim q_\phi}\left[\log q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})-\log p_\theta(\mathbf{z},\mathbf{Y}_{tr}|\mathbf{A},\mathbf{X})\right]\\
=&\,\mathbf{E}_{\mathbf{z}\sim q_\phi}\left[-(\log q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})-\log p(\mathbf{z}))\right]+\\
&\,\mathbf{E}_{\mathbf{z}\sim q_\phi}\left[\log p_\theta(\mathbf{z},\mathbf{Y}_{tr}|\mathbf{A},\mathbf{X})-\log p(\mathbf{z})\right]\\
=&-\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})||p(\mathbf{z})\right]+\\
&\,\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})}\left[\log p_\theta(\mathbf{Y}_{tr}|\mathbf{z},\mathbf{A},\mathbf{X})\right]\\
=&\,\mathcal{L}_{en}+\mathcal{L}_{de}
\end{aligned}
$$

The derived ELBO includes two terms, which respectively correspond to the encoder and decoder training of VR-GNN. The first term $\mathcal{L}_{en}$ is a KL divergence where the encoder ϕ is trained to generate relation vector $\mathbf{z}$ by observed graph information $\mathbf{A}$, $\mathbf{X}$, $\mathbf{Y}_{tr}$. Meanwhile, $\mathbf{z}$ is controlled by a manually assigned prior distribution $p(\mathbf{z})$. For the second term $\mathcal{L}_{de}$, the decoder θ is trained to employ generated $\mathbf{z}$ together with $\mathbf{A}$ and $\mathbf{X}$ to predict observed node labels $\mathbf{Y}_{tr}$.

In the inference phase, the learned encoder ϕ can be directly used to generate relation vectors, and the decoder θ is used to predict the unknown node labels. Hence we derive the final formulization of VR-GNN:

$$
p_{(\theta,\phi)}(\mathbf{Y}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})=\int q_\phi(\mathbf{z}|\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})\,p_\theta(\mathbf{Y}|\mathbf{z},\mathbf{A},\mathbf{X},\mathbf{Y}_{tr})\,\mathrm{d}\mathbf{z} \tag{5}
$$

The framework demonstration is given in figure 2.

4.2 Encoder: Relation Vector Generation

We model the relation vector of each connection with a independent identically distribution assumption. Given $\mathbf{A}$, $\mathbf{X}$ and $\mathbf{Y}_{tr}$, the variational posterior distribution

$q_\phi(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})$ and prior distribution $p(\mathbf{z})$ can be factorized as:

$$\begin{aligned} q_\phi(\mathbf{z}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) &= \prod_{e_{ij} \in \mathcal{E}} q_\phi(\mathbf{z}_{ij}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) \\ p(\mathbf{z}) &= \prod_{e_{ij} \in \mathcal{E}} p(\mathbf{z}_{ij}) \end{aligned} \tag{6}$$

and the learning object of encoder can be rewritten as:

$$\mathcal{L}_{en} = -\sum_{e_{ij} \in \mathcal{E}} \mathrm{KL}\left[q_\phi(\mathbf{z}_{ij}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) || p(\mathbf{z}_{ij})\right] \tag{7}$$

We let the posterior be a multivariate normal distribution, and the prior a standard multivariate normal distribution, which is flexible and could make the computation analytical:

$$\begin{aligned} q_\phi(\mathbf{z}_{ij}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) &= \mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{\sigma}^2_{ij}\mathbf{I}) \\ p(\mathbf{z}_{ij}) &= \mathcal{N}(\mathbf{0}, \mathbf{I}) \end{aligned} \tag{8}$$

where $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{\sigma}^2_{ij}$ are distribution parameters to be learned.

Furthermore, the relation vector $\mathbf{z}_{ij}$ expects to encode the comprehensive homophily and heterophily characteristics of a connection. To achieve this goal, we mainly consider three aspects of information to generate $\mathbf{z}_{ij}$: structure, feature and label (corresponding to $\mathbf{A}$, $\mathbf{X}$ and $\mathbf{Y}_{tr}$). It is a reasonable choice since there have been works showing that graph topology, node feature and node labels are served as important factors for homophily and heterophily modeling [19–21].

Specifically, we decompose $\mathbf{z}_{ij}$ into three sub-relation vectors generated from varying aspects, then linearly combined them together for fusing the information:

$$\mathbf{z}_{ij} = \alpha_s \mathbf{z}_{s,ij} + \alpha_f \mathbf{z}_{f,ij} + \alpha_l \mathbf{z}_{l,ij} \tag{9}$$

where $\alpha\cdot$ are hyper-parameters for composing weight. $\mathbf{z}_{s,ij}$, $\mathbf{z}_{f,ij}$ and $\mathbf{z}_{l,ij}$ denote structure, feature and label sub-relation vectors, which are set as multivariate normal distribution:

$$\begin{aligned} q_\phi(\mathbf{z}_{s,ij}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) &= q_\phi(\mathbf{z}_{s,ij}|\mathbf{A}) = \mathcal{N}(\boldsymbol{\mu}_{s,ij}, \boldsymbol{\sigma}^2_{s,ij}\mathbf{I}) \\ q_\phi(\mathbf{z}_{f,ij}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) &= q_\phi(\mathbf{z}_{f,ij}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}_{f,ij}, \boldsymbol{\sigma}^2_{f,ij}\mathbf{I}) \\ q_\phi(\mathbf{z}_{l,ij}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr}) &= q_\phi(\mathbf{z}_{l,ij}|\mathbf{Y}_{tr}) = \mathcal{N}(\boldsymbol{\mu}_{l,ij}, \boldsymbol{\sigma}^2_{l,ij}\mathbf{I}) \end{aligned} \tag{10}$$

Once we have got each sub-relation's expectation $\boldsymbol{\mu}_{\cdot,ij}$ and variance $\boldsymbol{\sigma}^2_{\cdot,ij}$, we can further derive the distribution of $\mathbf{z}_{ij}$ as:

$$\begin{aligned} \boldsymbol{\mu}_{ij} &= \alpha_s \boldsymbol{\mu}_{s,ij} + \alpha_f \boldsymbol{\mu}_{f,ij} + \alpha_l \boldsymbol{\mu}_{l,ij} \\ \boldsymbol{\sigma}^2_{ij} &= \alpha_s^2 \boldsymbol{\sigma}^2_{s,ij} + \alpha_f^2 \boldsymbol{\sigma}^2_{f,ij} + \alpha_l^2 \boldsymbol{\sigma}^2_{l,ij} \end{aligned} \tag{11}$$

and we detail the design of each sub-relation vector below.

Structure Sub-relation Vector

For capturing structure aspect information, we assign a randomly initialized expectation and variance embedding for each connection:

$$\begin{aligned} \boldsymbol{\mu}_{s,ij} &\in \mathbb{R}^{|\mathcal{E}|\times H} \\ \boldsymbol{\sigma}_{s,ij} &\in \mathbb{R}^{|\mathcal{E}|\times H} \end{aligned} \tag{12}$$

where H denotes hidden dimension. Because there is no guidance for generation, the subsequent learning of sub-relation vector is in fact based on the graph structure.

Feature Sub-relation Vector

This is motivated by the observation that the feature of edge endpoints can be regarded as weak label information and serve as an indicator for connecting homophily and heterophily [20]. Specifically, we employ the MLP to transform the concatenation of two endpoints feature, and generate expectation and variance as follows:

$$\begin{aligned} \mathbf{f}_{ij} &= \mathrm{ReLU}\left(\mathrm{MLP}([\mathbf{x}_i \| \mathbf{x}_j])\right) \\ \boldsymbol{\mu}_{f,ij} &= \mathrm{MLP}(\mathbf{f}_{ij}) \\ \boldsymbol{\sigma}_{f,ij} &= \mathrm{MLP}(\mathbf{f}_{ij}) \end{aligned} \tag{13}$$

Note that the above MLPs are different modules. For reducing used symbols, we adopt the same denotation (the same below).

Label Sub-relation Vector

The label of two end-nodes can provide direct homophily and heterophily description for a connection, but the usage of label information faces two problems: 1. There are only partially observed node labels; 2. The introduction of label information for in-degree node may lead to label leakage problem, as message passing will bring the "correct answer" to the node. Therefore we only use out-degree node label to generate sub-relation vector. Specifically, for a connection e_{ij}, we take node v_i's label y_i as one-hot vector, and feed it into MLP to generate expectation and variance embedding. For unobserved labels, we set the one-hot vector as zero. The process can be formulized as following:

$$\begin{aligned} \boldsymbol{\mu}_{l,ij} &= \mathrm{MLP}(y_i) \\ \boldsymbol{\sigma}_{l,ij} &= \mathrm{MLP}(y_i) \end{aligned} \tag{14}$$

Relation Vector Generation

After getting each sub-relations' mean and variance embedding, we combine them into the final embedding $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{\sigma}^2_{ij}$ by equation 11. Additionally, instead of directly sampling $\mathbf{z}_{ij}$, we apply the re-parameterization trick of VAE [17] to make the sampling process derivable:

$$\mathbf{z}_{ij} = \boldsymbol{\mu}_{ij} + \boldsymbol{\sigma}_{ij}\epsilon \tag{15}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. After getting each edge's posterior $\mathbf{z}_{ij}$, equation 7 is conducted to calculate the encoder loss.

4.3 Decoder: Message Passing with Relation Vector

The decoder aims to incorporate generated relation vectors into message-passing framework, to complete downstream node classification task. In this work, our inspiration is mainly based on the idea of TransE [15], a classical knowledge graph model, that the relation both serves as a semantic and numerical translation for connected nodes. Our message-passing function can be formalized as follows:

$$\begin{gathered}\mathbf{h}_i^{l+1} = \mathrm{U}(\mathbf{h}_i^l, \mathrm{AGG}(\varphi(\mathbf{h}_j^l, \mathbf{z}_{ji}^l) : j \in \mathcal{N}_i)) \\ \varphi(\mathbf{h}_j^l, \mathbf{z}_{ji}^l) = \mathbf{W}^l \mathbf{h}_j^l + \mathbf{z}_{ji}^l\end{gathered} \tag{16}$$

where $\mathbf{h}_i^l$ denotes the embedding of node v_i in layer l; $\mathbf{z}_{ji}^l$ denotes the relation vector of edge e_{ji} in layer l, with $\mathbf{z}_{ji}^0 = \mathbf{z}_{ji}$. For each neighbor, we apply a matrix transformation and relation translation before aggregating. This can convert neighbors to a more proper feature with the central node, and flexibly model the homophily and heterophily property of each connection when message passing.

After each layer, relation vectors will also go through a matrix transformation to maintain the layer consistency with node embedding:

$$\mathbf{z}_{ji}^{l+1} = \mathbf{W}_{rel}^l \mathbf{z}_{ji}^l \tag{17}$$

Next, we give the overall procedure of the decoder and corresponding implementation details. Firstly, we apply MLP to transform original node feature to higher-level embedding:

$$\mathbf{h}_i^0 = \mathrm{ReLU}(\mathrm{MLP}(\mathbf{x}_i)) \tag{18}$$

Secondly, we conduct aggregation function. Considering that attention mechanism can adaptively model the influence of different nodes, we take self-attention [22] to aggregate neighbor information:

$$\bar{\mathbf{h}}_i^l = \mathrm{AGG}(\varphi(\mathbf{h}_j^l, \mathbf{z}_{ji}^l) : j \in \mathcal{N}_i) = \sum_{j \in \mathcal{N}_i} \beta_{ij}^l (\varphi(\mathbf{h}_j^l, \mathbf{z}_{ji}^l)) \tag{19}$$

where β_{ij}^l is attention coefficient:

$$\beta_{ij}^l = \frac{\exp\{\mathbf{h}_i^l \varphi(\mathbf{h}_j^l, \mathbf{z}_{ji}^l)\}}{\sum_{k \in \mathcal{N}_i} \exp\{\mathbf{h}_i^l \varphi(\mathbf{h}_k^l, \mathbf{z}_{ki}^l)\}} \tag{20}$$

Thirdly, we conduct updating function to renew the central node embedding:

$$\mathbf{h}_i^{l+1} = \theta \bar{\mathbf{h}}_i^l + (1 - \theta) \mathbf{h}_i^0 \tag{21}$$

where θ is to balance $\bar{\mathbf{h}}_i^l$ and $\mathbf{h}_i^0$, that can maintain the computing stability by attaching a residual of initial layer. Then the second and third steps will iterate L times to get the output node representation:

$$\mathbf{h}_i = \mathbf{h}_i^L \tag{22}$$

Finally, we employ an MLP to perform node classification:

$$y_i^{pred} = \mathrm{MLP}(\mathbf{h}_i) \tag{23}$$

4.4 Training and Inference

Training

After getting the prediction of each node, we can calculate a semi-supervised loss for the decoder, which corresponds to the second term of equation 4:

$$\mathcal{L}_{de} = -\frac{1}{N_{tr}} \sum_{v_i} \mathrm{CE}(y_i^{pred}, y_i) \tag{24}$$

where N_{tr} is the training node number; $\mathrm{CE}(\cdot)$ denotes the cross entropy function. Then with the encoder loss of equation 7, we could derive the overall loss of the model:

$$\mathcal{L}_{(\theta,\phi)} = \gamma \mathcal{L}_{en} + (1-\gamma)\mathcal{L}_{de} \tag{25}$$

Here we add a weighting hyper-parameter γ between the encoder and decoder, which aims to provide training process a more flexible focus.

Inference

In the inference phase, when generating $\mathbf{z}_{ij}$ by $q_\phi(\mathbf{z}_{ij}|\mathbf{A}, \mathbf{X}, \mathbf{Y}_{tr})$, we directly use the expectation $\boldsymbol{\mu}_{ij}$ as the relation vector of edge e_{ij}. We ignore the variance $\boldsymbol{\sigma}_{ij}$ to reduce the noise for inference, which is similar as [23].

4.5 Algorithm Complexity Analysis

Here we analyse the time complexity of VR-GNN. The generation of each sub-relation vector costs $\mathcal{O}(|\mathcal{E}|)$, hence the time complexity of encoder is $\mathcal{O}(|\mathcal{E}|)$. For the decoder, the feature transformation in equation 18 is of $\mathcal{O}(|\mathcal{V}|)$ complexity. For each GNN layer, the aggregation function and updating function respectively cost $\mathcal{O}(|\mathcal{E}|)$ and $\mathcal{O}(|\mathcal{V}|)$ complexity. Finally, the MLP for classification is of $\mathcal{O}(|\mathcal{V}|)$ complexity.

Therefore, the overall time complexity of VR-GNN is:

$$\mathcal{O}(L|\mathcal{E}| + L|\mathcal{V}|)$$

where L is the layer number. This matches the complexity degree of other GNN baselines, like FAGCN [12] and GPR-GNN [24].

5 Experiments

5.1 Experiment Setup

Table 2: Homophily Datasets Statistics

	Cora	CiteSeer	PubMed
# Node	2708	3327	19717
# Edge	10556	9228	88651
# Feature	1433	3703	500
# Class	7	6	5
Homophily Ratio $\mathcal{H}$	0.656	0.578	0.644

Table 3: Heterophily Datasets Statistics

	Chameleon	Squirrel	Actor	Texas	Cornell
# Node	2277	5201	7600	183	183
# Edge	62742	396706	53318	558	554
# Feature	767	2089	932	1703	1703
# Class	5	5	5	5	5
Homophily Ratio $\mathcal{H}$	0.024	0.055	0.008	0.016	0.137

Datasets

In the experiments, we utilize eight real-world datasets with different homophily ratios:

- Cora, Citeseer and Pubmed are three citation networks [25, 26] with high homophily ratio.
- Chameleon and Squirrel [10] are page-page networks extracted from Wikipedia of specific topics, with low homophily ratio.
- Actor [10] is constructed according to the actor co-occurrence in Wikipedia pages and holds low homophily ratio.
- Cornell and Texas [10] are two sub-datasets of WebKB, a webpage database constructed by Carnegie Mellon University, and possess low homophily ratio.

In practice, Cora, Citeseer and Pubmed are regarded as homophilic graphs, and Chameleon, Squirrel, Actor, Cornell and Texas are considered as heterophilic graphs. The statistics of datasets are demonstrated in table 2 and tabel 3. We use the same dataset partition as [10], which randomly splits nodes of each class into train/validation/test set with a ratio of 60%/20%/20%.

Baselines

We compare VR-GNN with several existing strong baselines to verify the effectiveness of our method, including:

- **Non-GNN Method**: **MLP**: that only considers node features and ignores graph structure;
- **Homophily-based GNNs**: **GCN** [4], **GAT** [2] and **SGC** [27], which are designed with the homophily assumption;
- **Heterophily-based GNNs**: **FAGCN** [12], **GPR-GNN** [24], **BernNet**[13], **ACM-GCN** [14], **GeomGCN** [10], **H$_2$GCN** [28], **HOC-GCN** [29], **BM-GCN** [11], **GloGNN++** [30], **Auto-HeG** [31], which are designed with the heterophily assumption.

For MLP, GCN, GAT and SGC models, we tune them for the optimal parameters. For FAGCN, GPR-GNN, ACM-GCN, BM-GCN and GloGNN++, we rerun the models with the default parameters given by the author. For GeomGCN, H$_2$GCN, BernNet and HOC-GCN, Auto-HeG, we report the results of published papers. For all methods, we report the mean accuracy with a 95% confidence interval of 10 runs.

For our method, we use early stopping strategy with 200 epochs and set an maximum epoch number as 1000. We set the dimensional size of node embedding and relation vector as 64, and the layer number of GNN as 2. We use Adam optimizer to train the model, and tune learning rate from $\{0.01, 0.02, 0.05, 0.001, 0.002, 0.005\}$, weight decay from $\{0, 1e-4, 5e-4, 1e-5, 5e-5\}$. We tune the hyperparameters $\alpha_s, \alpha_f, \alpha_l, \theta, \gamma$ from 0 to 1, with 0.1 step size. To mitigate the overfitting problem, we take dropout when training. We implement our method based on PyTorch Geometric (PyG) library [32] and Python 3.9.12. The program is executed on 32GB Tesla V100 GPU.

5.2 Results of Node classification Task

Table 4 and 5 list the results of VR-GNN and other baselines for node classification task, from which we can observe that:

VR-GNN outperforms all the other methods on all five heterophilic datasets. This proves the effectiveness of employing relation vectors to achieve a heterophily-based GNN. Specifically, VR-GNN significantly outperforms traditional GNNs, i.e. GCN, GAT and SGC, relatively by 24.8%, 20.5% and 41.3% on average, since they cannot generalize to heterophily scenarios. Compared with other heterophily-based GNNs, including both the method type of mixing high-order neighbors and passing signed messages, VR-GNN also achieves effective improvements, like 5.1% over ACM-GNN on Chameleon, 11.8% over BernNet on Squirrel, 13.8% over GloGNN++ on Actor, 14.1% over BM-GCN on Texas, 4.3% over GPR-GNN on Cornell. These results demonstrate that VR-GNN could model the heterophilic connections more flexibly and expressively, meanwhile without destroying the graph structure.

On homophilic datasets, i.e. Cora, Citeseer and Pubmed, VR-GNN performs better or comparably to the baselines. Specifically, VR-GNN outperforms all the methods on Citeseer and Pubmed dataset. For Cora dataset, VR-GNN also achieves the second best result with only 0.87 difference with GAT. These show that VR-GNN possesses

Table 4: Results on homophilic with mean accuracy (%) ± **95**% confidence interval. The best and second best results are in bold and underlined. OOM means out of memory when reproducing. H_2GCN and HOC-GCN report mean accuracy (%)± standard deviation.

	Homophilic Datasets		
	Cora	Citeseer	Pubmed
MLP	$77.83_{\pm 1.28}$	$76.77_{\pm 0.90}$	$85.67_{\pm 0.33}$
GCN	$87.87_{\pm 1.03}$	$80.26_{\pm 0.60}$	$87.16_{\pm 0.27}$
GAT	$\mathbf{89.14}_{\pm 0.95}$	$81.45_{\pm 0.59}$	$87.51_{\pm 0.25}$
SGC	$86.78_{\pm 0.95}$	$80.71_{\pm 0.55}$	$81.93_{\pm 0.21}$
FAGCN	$87.58_{\pm 1.09}$	$\underline{81.79_{\pm 1.01}}$	$84.26_{\pm 0.41}$
GPR-GNN	$88.11_{\pm 1.05}$	$79.51_{\pm 0.85}$	$89.25_{\pm 0.46}$
BernNet	$88.52_{\pm 0.95}$	$80.09_{\pm 0.79}$	$88.48_{\pm 0.41}$
ACM-GCN	$88.01_{\pm 0.68}$	$80.87_{\pm 0.81}$	$89.20_{\pm 0.20}$
GeomGCN	$85.4_{\pm 0.26}$	$76.42_{\pm 0.37}$	$88.51_{\pm 0.08}$
H_2GCN	$86.92_{\pm 1.35}$	$77.07_{\pm 1.64}$	$89.40_{\pm 0.34}$
HOC-GCN	$87.04_{\pm 1.10}$	$76.15_{\pm 1.88}$	$88.79_{\pm 0.40}$
BM-GCN	$87.53_{\pm 0.70}$	$80.29_{\pm 1.02}$	$\underline{89.32_{\pm 0.47}}$
GloGNN++	$76.85_{\pm 0.64}$	$75.33_{\pm 0.78}$	OOM
Auto-HeG	$86.88_{\pm 1.10}$	$75.81_{\pm 1.52}$	$89.29_{\pm 0.27}$
VR-GNN	$\underline{88.27_{\pm 0.89}}$	$\mathbf{81.95}_{\pm 0.77}$	$\mathbf{89.65}_{\pm 0.33}$

a consistent performance on homophily scenarios, which further proves the adaptive modeling capacity of relation vectors.

5.3 Ablation Study of Three Sub-relations

To evaluate the effect of each sub-relation vector part, we conduct the ablation study of only removing one sub-relation part and simultaneously removing two of them. We take Chameleon, Squirrel, Texas and Citeseer as example datasets. The results are demonstrated in table 6. The subscripts s, f, l respectively denote the sub-relation used. We can see that generating relation vectors with absence of some relation cannot provide stable performance across datasets compared to VR-GNN, which verifies the necessity of composing all three parts.

5.4 Compared with Other Encoder Designs

Currently, many works use GNN as an encoder to get the hidden embedding of data [23, 33]. To further verify the efficacy of our encoder design, we compare it with some GNN-based encoders such as GCN and SGC on four datasets. Specifically, we first employ GCN (SGC) to get node embeddings, then concatenate the embeddings of two endpoints of each edge, and use MLP to calculate the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$ of the relation vector. Other structure of VR-GNN remains the same. The results are shown in table 7. We can observe that our encoder outperforms GCN and SGC by 3.1% and 2.5% on average. Our method is designed to explicitly extract three aspects of

Table 5: Results on heterophilic datasets with mean accuracy (%)$\pm$**95**% confidence interval. The best and second best results are in bold and underlined. H$_2$GCN and HOC-GCN report mean accuracy (%)$\pm$ standard deviation.

	Heterophilic Datasets				
	Chameleon	Squirrel	Actor	Texas	Cornell
MLP	47.61$_{\pm 1.23}$	31.73$_{\pm 0.98}$	39.20$_{\pm 0.82}$	89.51$_{\pm 1.80}$	89.51$_{\pm 2.60}$
GCN	62.93$_{\pm 1.82}$	46.33$_{\pm 1.06}$	33.73$_{\pm 0.85}$	78.69$_{\pm 3.28}$	65.74$_{\pm 4.43}$
GAT	63.26$_{\pm 1.40}$	42.81$_{\pm 1.14}$	35.93$_{\pm 0.42}$	79.67$_{\pm 2.30}$	77.70$_{\pm 2.62}$
SGC	64.55$_{\pm 1.36}$	40.45$_{\pm 0.71}$	29.97$_{\pm 0.66}$	69.18$_{\pm 2.62}$	52.62$_{\pm 3.61}$
FAGCN	63.30$_{\pm 1.08}$	41.26$_{\pm 1.24}$	38.36$_{\pm 0.72}$	90.00$_{\pm 3.78}$	88.38$_{\pm 2.16}$
GPR-GNN	66.43$_{\pm 0.74}$	52.96$_{\pm 0.92}$	39.69$_{\pm 0.72}$	91.80$_{\pm 1.64}$	88.85$_{\pm 2.13}$
BernNet	68.29$_{\pm 1.58}$	51.35$_{\pm 0.73}$	<u>41.79$_{\pm 1.01}$</u>	<u>93.12$_{\pm 0.65}$</u>	<u>92.13$_{\pm 1.64}$</u>
ACM-GCN	67.74$_{\pm 1.39}$	<u>53.59$_{\pm 0.70}$</u>	39.86$_{\pm 1.00}$	92.97$_{\pm 2.43}$	91.16$_{\pm 1,62}$
GeomGCN	61.06$_{\pm 0.49}$	38.28$_{\pm 0.27}$	31.81$_{\pm 0.24}$	58.56$_{\pm 1.77}$	55.59$_{\pm 1.59}$
H$_2$GCN	57.11$_{\pm 1.58}$	36.42 $_{\pm 1.89}$	35.86$_{\pm 1.03}$	84.86$_{\pm 6.77}$	82.16$_{\pm 4.80}$
HOC-GCN	-	-	36.82$_{\pm 0.84}$	85.17$_{\pm 4.40}$	84.32$_{\pm 4.32}$
BM-GCN	<u>69.85$_{\pm 0.85}$</u>	51.59$_{\pm 1.05}$	39.23$_{\pm 0.70}$	83.11$_{\pm 2.79}$	82.79$_{\pm 2.95}$
GloGNN++	69.58$_{\pm 1.16}$	48.83$_{\pm 0.69}$	37.06$_{\pm 0.46}$	82.79$_{\pm 2.46}$	82.13$_{\pm 2.62}$
Auto-HeG	-	-	37.43$_{\pm 1.37}$	86.76$_{\pm 4.60}$	83.51$_{\pm 6.56}$
VR-GNN	**71.21**$_{\pm 1.17}$	**57.50**$_{\pm 1.18}$	**42.16**$_{\pm 0.42}$	**94.86**$_{\pm 1.89}$	**92.70**$_{\pm 2.70}$

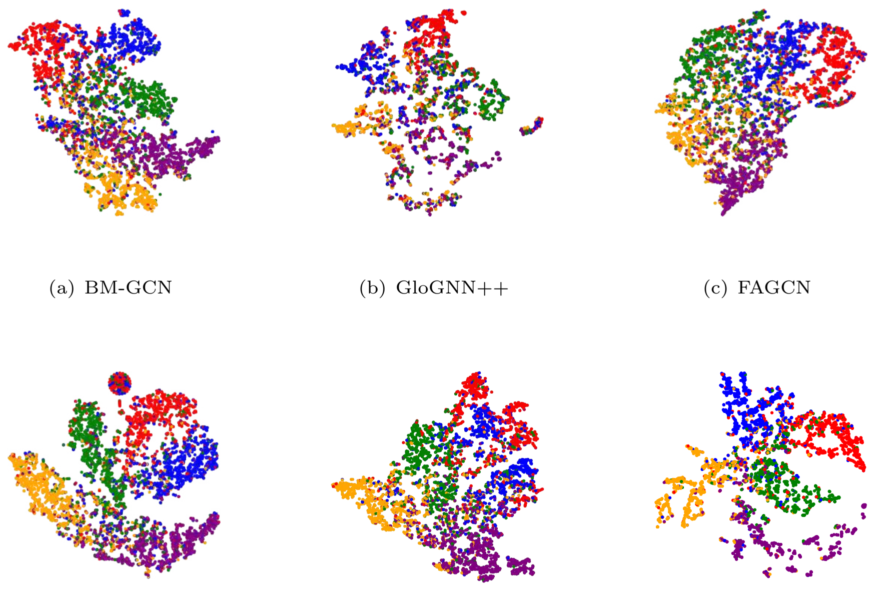
Table 6: Ablation study of three sub-relations.

Datasets	Chameleon	Squirrel	Texas	Citeseer
VR-GNN$_s$	69.89	53.37	93.24	81.02
VR-GNN$_f$	69.54	52.74	93.51	81.16
VR-GNN$_l$	70.00	53.43	92.97	80.96
VR-GNN$_{sf}$	70.22	55.12	93.71	81.36
VR-GNN$_{sl}$	70.69	56.78	93.71	81.20
VR-GNN$_{fl}$	70.30	54.48	93.51	81.88
VR-GNN	**71.21**	**57.50**	**94.86**	**81.95**

information for generation, which can make the encoding process more effective and less noisy.

5.5 Visualization Analysis

To show the modeling effect of VR-GNN more intuitively, we conduct the node embedding visualization for Squirrel dataset. We extract the node embedding of VR-GNN and five promising baselines (BM-GCN, GloGNN++, FAGCN, GPR-GNN and ACM-GCN), then employ t-SNE [34] algorithm to map them into 2-dimensional space for visualization. The results are shown in figure 3. We can observe that VR-GNN learns more discriminative node embedding, which is more cohesive within the same category and dispersed between the different categories. This further proves the validity of



(a) BM-GCN (b) GloGNN++ (c) FAGCN

(d) GPR-GNN (e) ACM-GCN (f) VR-GNN

Fig. 3: Node embedding visualization for Squirrel dataset. Different colors correspond to different node classes.

Table 7: Comparison results with other encoder designs.

Encoder	Datasets			
	Chameleon	Squirrel	Texas	Citeseer
GCN	69.38	52.76	91.35	80.68
SGC	69.93	53.75	91.08	80.78
Ours	**71.21**	**57.50**	**94.86**	**81.95**

the relation vector based message-passing, which can produce more accurate assimilation and dissimilation effects between nodes according to homophily and heterophily connections.

5.6 Hyper-parameter Analysis

In this section, we investigate the sensitivity of hyper-parameters used in VR-GNN. We take Chameleon, Squirrel, Cora and Pubmed as example datasets.

Weight Parameter γ.

To investigate the influence of KL divergence loss (encoder loss) for learning effect, we conduct the sensitivity experiment for parameter γ (equation 25). We test the node classification accuracy of VR-GNN with γ ranging from 0.1 to 0.9. The results are reported in figure 4. We can discover that the trends of γ are the same in all datasets where a low point slowly rise to a maximum and then gradually decline. This is because KL loss restricts the generated relation vector not to deviate far from the prior distribution, and the small γ may lead to too large or too small embedding, while the large γ will harm the learning of classification task.

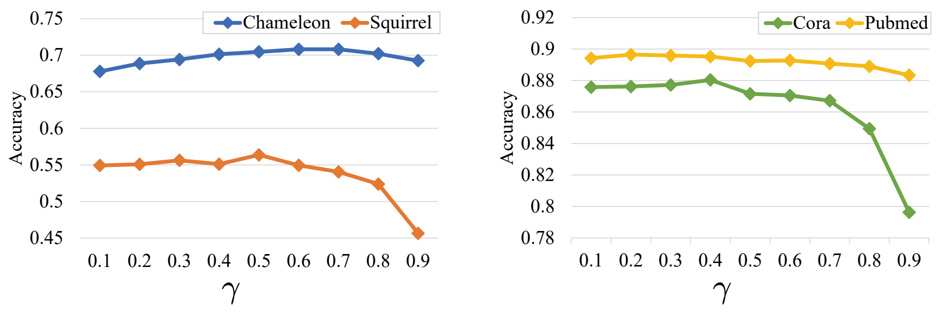


Fig. 4: Influence of weight parameter γ on Chameleon, Squirrel, Cora and Pubmed dataset.

Weight Parameter θ.

The parameter θ balances the node's original feature and neighbor aggregation information (equation 21). A larger θ indicates a greater role graph structure plays. We test the node classification accuracy of VR-GNN with θ from 0.1 to 0.9. The results are shown in figure 5. We can observe that VR-GNN has greater θ on Chameleon and Squirrel datasets. This is because in Chameleon and Squirrel graph structure is more important, while in Cora and Pubmed node feature is more important. This can also be proven in table 4 and table 5: VR-GNN improves 49.6% and 81.2% over MLP on Chameleon and Squirrel, while only 13.4% and 4.6% on Cora and Pubmed.

Analysis of Parameters $\alpha_s, \alpha_f, \alpha_l$.

Figure 6 shows the best results of α_s, α_f and α_l on four datasets. We can observe that although the parameters of different datasets are not completely consistent, they still show some similarity, which is related to the characteristics of corresponding datasets. Previous work [35] has shown that the label distribution of Chameleon and Squirrel greatly improves the results, so the weights of α_l are the largest on these two datasets. Cora and PubMed have better initial feature, so α_f has more weight.This further illustrates the necessity of combining all three relations to achieve stable performance across datasets.

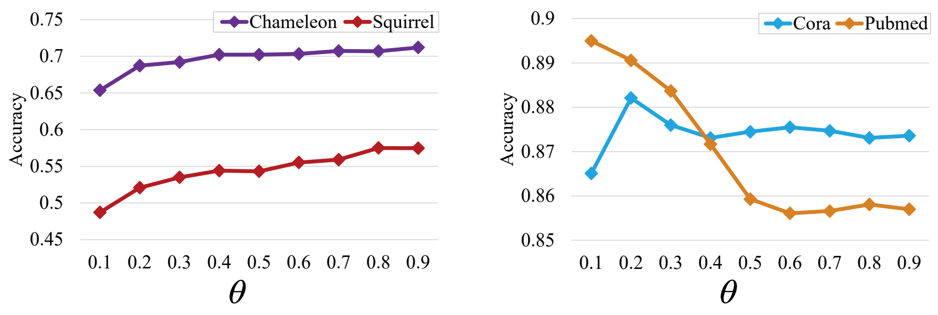


Fig. 5: Influence of weight parameter θ on Chameleon, Squirrel, Cora and Pubmed dataset.

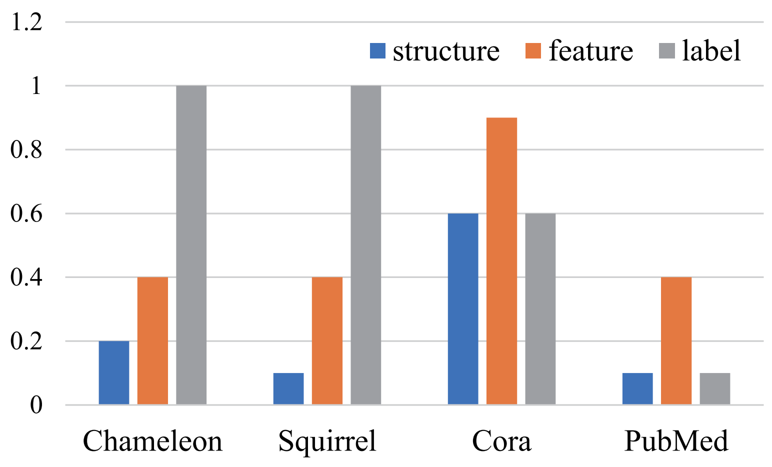


Fig. 6: Analysis of parameters $\alpha_s, \alpha_f, \alpha_l$.

5.7 Visualization Analysis of Relation Vectors

In this section, we give some intuitive demonstrations of the relation vector and compare it with the signed message method, which models the heterophilic edge as a high frequency signal and is the mainstream of current methods.

For the message passing on edge e_{ji}, we denote the relation vector message as $\mathbf{m}_{ji}^{rel}$ and compute it according to equation 16:

$$\mathbf{m}_{ji}^{rel} = \mathbf{W}\mathbf{h}_j + \mathbf{z}_{ji}$$

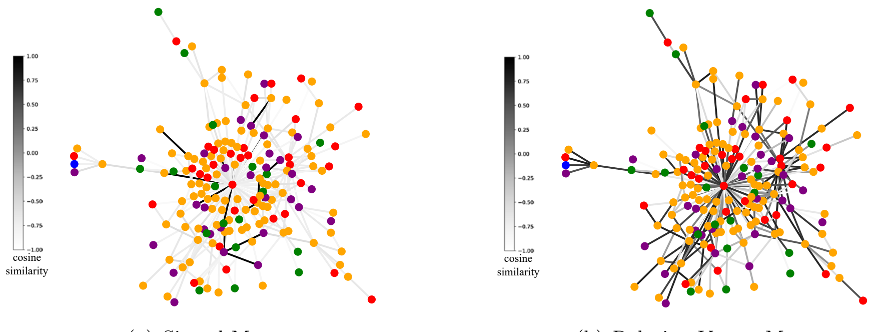


Fig. 7: Similarity visualization between the message passed and the central node class. Different node colors represent different classes. The edge shade indicates the cosine similarity value from -1 to 1.

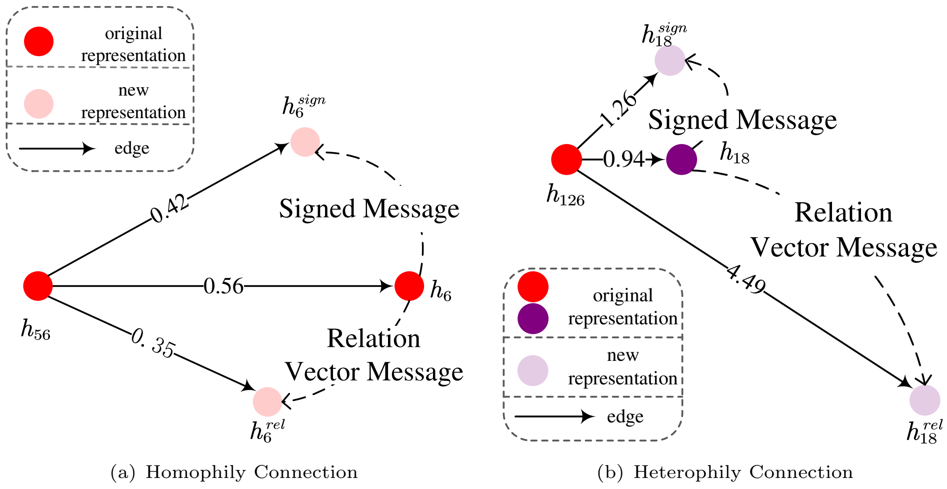


Fig. 8: Relation vector message and signed message for assimilating and dissimilating nodes. The two edges are $e_{56,6}$ and $e_{126,18}$ in Texas dataset. Red and purple colors denote the node classes, and light coloring represents the new representation after "aggregation". The number of edges indicate the Euclidean distance of two nodes.

For the signed message, we directly assign the correct sign for the connection, with $+1$ for homophily and -1 for heterophily:

$$\mathbf{m}_{ji}^{sign} = \begin{cases} \mathbf{W}\mathbf{h}_j, & y_j = y_i \\ -\mathbf{W}\mathbf{h}_j, & y_j \neq y_i \end{cases}$$

where matrix $\mathbf{W}$ is the same in $\mathbf{m}_{ji}^{rel}$ and $\mathbf{m}_{ji}^{sign}$. Then we conduct two aspects of analysis, to compare the effect of two messages for modeling homophily/heterophily connections and helping node classification task.

Effect for Modeling Homophily/Heterophily

Modeling homophily/heterophily property of a connection means that the message could conduct effective assimilating/dissimilating operation between connected nodes. To evaluate this, we utilize the embedding of VR-GNN in $L-1$ layer and compute the relation vector message $\mathbf{m}_{ji}^{rel}$ and signed message $\mathbf{m}_{ji}^{sign}$ for each edge. Then by adding the messages to $\mathbf{h}_i$ respectively, we can get the new representation of central node $\mathbf{h}_i^{rel}$ and $\mathbf{h}_i^{sign}$. We compare $\mathbf{h}_i^{rel}$, $\mathbf{h}_i^{sign}$ and original representation $\mathbf{h}_i$ with the neighbor $\mathbf{h}_j$, to evaluate the distance change. For demonstration, we take two edges of Texas dataset as the example. The results are shown in figure 8. We can see that the "aggregation" of relation vector message makes the connected nodes closer under homophily and more distant under heterophily, which shows its superiority over signed message method.

Effect for Helping Node Classification

In addition to accurately describe the connection property, a valid message should also be consistent with the central node class, so as to facilitate the node classification. To show this, we first calculate the mean center for each node class, based on L layer's node embedding of VR-GNN. Then we compute the message $\mathbf{m}_{ji}^{rel}$ and $\mathbf{m}_{ji}^{sign}$ of $L-1$ layer. For each edge, we compare $\mathbf{m}_{ji}^{rel}$ and $\mathbf{m}_{ji}^{sign}$ with the class center of node v_i using cosine similarity. The results are shown in figure 7. We can observe that relation vector messages are more similar with the central node class than signed messages on most edges, which achieves a more center-cohesive node representation for classification and corresponds to the node visualization results in figure 3.

6 Related Work

GNNs have been established as powerful and flexible tools for graph representation learning. A large class of GNNs build multilayer models, where each layer operates on the previous layer to generate new representations using a message-passing mechanism to aggregate local neighborhood information. According to the connection characteristics of the applied graphs, we introduce two families of GNN works: homophily-based GNNs and heterophily-based GNNs.

6.1 Homophily-based GNNs

The homophily-based GNNs are developed mainly for homophilic graphs, which are the earliest proposed and extensively studied. It contains two major types: spectral-based GNNs and spacial-base GNNs. Spectral-based GNNs use spectral graph theory to design aggregation strategy: GCN [4] propose the most common graph convolution mode with simplifying the polynomial convolution kernel and renormalization trick; SGC [27] emphasizes that the entanglement of convolutional filters and weight matrices may be harmful to the performance of GNNs and propose a lightweight GNN model. Spatial-based GNNs design the GNN aggregation mechanism based directly on the connections of graph: GAT [2] applies the self-attention mechanism to adaptively adjust aggregation weights; GraphSage [36] proposes an inductive node embedding method with sampling and aggregating features from neighborhood. GIN [37] designs a simple yet effective convolution mechanism to explore the upper bound of message-passing based GNNs under homophily. These methods directly use the original neighbor information to enhance node representation, which perform well on homophilic graphs, where connected nodes tend to possess similar features and belong to the same class. However, these methods suffer from poor performance and even underperform MLP that completely ignores the graph structure on heterophilic graphs where connected nodes have dissimilar features and different labels.

6.2 Heterophily-based GNNs

The poor performance of homophily-based GNNs on heterophilic scenarios inspires the study of heterophily-based GNNs. Heterophily-based GNNs can be mainly divided into two families based on designing methodologies: mixing high-order neighbors and passing signed messages. Mixing high-order neighbors expects to aggregate more homophilic nodes and remove heterophilic nodes. Specific introduction is as follows: Geom-GCN [10] proposes a novel geometric aggregation scheme to acquire more homophilic neighbors; BM-GCN [11] explores block-guided neighbors and conducts classified aggregation for both homophilic and heterophilic nodes; GloGNN [30] learns a coefficient matrix from graph and utilizes it to aggregate nodes with global homophily. HOC-GCN [29] incorporate a learnable homophily degree matrix into graph convolution framework for modeling the homophily and heterophily of networks. H_2GCN [28] applies some key designs, such as higher-order neighborhoods aggregation and combination of intermediate representations, to boost learning from graph with heterophily. However, these methods require specific priors to rewrite the graph structure, and such priors may not be applicable to all cases. Passing signed messages uses positive and negative signs to modify neighbor information. Specific introduction is as follows: FAGCN [12] proposes an adaptive method to capture both low and high-frequency graph signal by passing signed message; GPR-GNN [24] learns signed weighting of different orders of graph structure to deal with both homophily and heterophily. ACM-GCN [14] proposed a multi-channel mixing mechanism to enable adaptive filtering at different nodes. However, the single numerical sign suffers from limited expressing capacity, which causes the modeling to be inflexible and insufficient. In summary, current methods are faced with the problem of insufficient aggregation of neighbor information and inflexible aggregation mode, which is difficult to fully model homophily and heterophily.

7 Conclusion

In this paper, we investigate the influence of fine-grained edge semantic information on the node representation of heterophily graphs and propose a novel idea that introduces the concept of relation vector into the message-passing framework for modeling the implicit semantic of each edge. Based on the initial experimental verification of the effectiveness of relation vector, we further present Variational Relation Vector Graph Neural Network (VR-GNN), where relations are modeled as latent variable and inferred by three kinds of graph data, and the downstream node classification task is treated as the generation process guided by the learned latent variable. We show that VR-GNN can serve as a more effective way for relation inferring and applying, and extensive experiments on both homophily and heterophily datasets validate the superiority of our method.

References

[1] Wang, H., Xu, T., Liu, Q., Lian, D., Chen, E., Du, D., Wu, H., Su, W.: Mcne: an end-to-end framework for learning multiple conditional network representations of social network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1064–1072 (2019)

[2] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. International Conference on Learning Representations (2018). accepted as poster

[3] Sanyal, S., Anishchenko, I., Dagar, A., Baker, D., Talukdar, P.: Proteingcn: Protein model quality assessment using graph convolutional networks. bioRxiv (2020) https://doi.org/10.1101/2020.04.06.028266

[4] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)

[5] Klicpera, J., Bojchevski, A., Günnemann, S.: Predict then propagate: Graph neural networks meet personalized pagerank. In: International Conference on Learning Representations (2018)

[6] Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional networks. In: International Conference on Machine Learning, pp. 1725–1735 (2020). PMLR

[7] McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual Review of Sociology **27**(1), 415–444 (2001) https://doi.org/10.1146/annurev.soc.27.1.415 https://doi.org/10.1146/annurev.soc.27.1.415

[8] Lim, D., Hohne, F., Li, X., Huang, S.L., Gupta, V., Bhalerao, O., Lim, S.N.: Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. Advances in Neural Information Processing Systems **34** (2021)

[9] Li, Q., Han, Z., Wu, X.-M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

[10] Pei, H., Wei, B., Chang, K.C.-C., Lei, Y., Yang, B.: Geom-gcn: Geometric graph convolutional networks. In: International Conference on Learning Representations (2019)

[11] He, D., Liang, C., Liu, H., Wen, M., Jiao, P., Feng, Z.: Block modeling-guided graph convolutional neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 4022–4029 (2022)

[12] Bo, D., Wang, X., Shi, C., Shen, H.: Beyond low-frequency information in graph convolutional networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 3950–3957 (2021)

[13] He, M., Wei, Z., Huang, Z., Xu, H.: Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In: NeurIPS (2021)

[14] Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., Chang, X.-W., Precup, D.: Revisiting heterophily for graph neural networks. Conference on Neural Information Processing Systems (2022)

[15] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013)

[16] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272 (2017). PMLR

[17] Kingma, D.P., Welling, M.: Auto-Encoding Variational Bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings (2014)

[18] Mnih, A., Gregor, K.: Neural variational inference and learning in belief networks. In: International Conference on Machine Learning, pp. 1791–1799 (2014). PMLR

[19] Suresh, S., Budde, V., Neville, J., Li, P., Ma, J.: Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1541–1551 (2021)

[20] Yang, L., Li, M., Liu, L., Wang, C., Cao, X., Guo, Y., et al.: Diverse message passing for attribute with heterophily. Advances in Neural Information Processing Systems **34** (2021)

[21] Zhu, J., Rossi, R.A., Rao, A., Mai, T., Lipka, N., Ahmed, N.K., Koutra, D.: Graph neural networks with heterophily. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11168–11176 (2021)

[22] Bahdanau, D., Cho, K., Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. (2015). 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015

[23] Kipf, T.N., Welling, M.: Variational graph auto-encoders. NIPS Workshop on Bayesian Deep Learning (2016)

[24] Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized pagerank graph neural network. In: International Conference on Learning Representations (2020)

[25] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI magazine **29**(3), 93–93 (2008)

[26] Namata, G., London, B., Getoor, L., Huang, B., EDU, U.: Query-driven active surveying for collective classification. In: 10th International Workshop on Mining and Learning with Graphs, vol. 8, p. 1 (2012)

[27] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: International Conference on Machine Learning, pp. 6861–6871 (2019). PMLR

[28] Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Beyond homophily in graph neural networks: Current limitations and effective designs. Advances in Neural Information Processing Systems **33**, 7793–7804 (2020)

[29] Wang, T., Jin, D., Wang, R., He, D., Huang, Y.: Powerful graph convolutional networks with adaptive propagation mechanism for homophily and heterophily. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 4210–4218 (2022)

[30] Li, X., Zhu, R., Cheng, Y., Shan, C., Luo, S., Li, D., Qian, W.: Finding global homophily in graph neural networks when meeting heterophily. arXiv preprint arXiv:2205.07308 (2022)

[31] Zheng, X., Zhang, M., Chen, C., Zhang, Q., Zhou, C., Pan, S.: Autoheg: Automated graph neural network on heterophilic graphs. arXiv preprint arXiv:2302.12357 (2023)

[32] Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)

[33] Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., Zemel, R.: Neural relational inference for interacting systems. In: International Conference on Machine Learning, pp. 2688–2697 (2018). PMLR

[34] Hinton, G., Maaten, L.: Visualizing data using t-sne journal of machine learning research (2008)

[35] Ma, Y., Liu, X., Shah, N., Tang, J.: Is homophily a necessity for graph neural networks? In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, ??? (2022). https://openreview.net/forum?id=ucASPPD9GKN

[36] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)

[37] Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2018)

8 Declarations

8.1 Ethical Approval

Not applicable.

8.2 Funding

This work is supported by the National Key Research and Development Program of China (NO.2022YFB3102200) and Strategic Priority Research Program of the Chinese Academy of Sciences with No. XDC02030400.

8.3 Availability of data and materials

All datasets are available in PyTorch Geometric (PyG) library [32].