Relation Embedding based Graph Neural Networks for Handling Heterogeneous Graph

Junfu Wang  
School of Computer Science and Engineering, Beihang University  
Beijing, China  
wangjunfu@buaa.edu.cn

Yuanfang Guo  
School of Computer Science and Engineering, Beihang University  
Beijing, China  
andyguo@buaa.edu.cn

Liang Yang  
School of Artificial Intelligence, Hebei University of Technology  
Tianjin, China  
yangliang@vip.qq.com

Yunhong Wang  
School of Computer Science and Engineering, Beihang University  
Beijing, China  
yhwang@buaa.edu.cn

ABSTRACT

Heterogeneous graph learning has drawn significant attentions in recent years, due to the success of graph neural networks (GNNs) and the broad applications of heterogeneous information networks. Various heterogeneous graph neural networks have been proposed to generalize GNNs for processing the heterogeneous graphs. Unfortunately, these approaches model the heterogeneity via various complicated modules. This paper aims to propose a simple yet efficient framework to make the homogeneous GNNs have adequate ability to handle heterogeneous graphs. Specifically, we propose Relation Embedding based Graph Neural Networks (RE-GNNs), which employ only one parameter per relation to embed the importance of edge type relations and self-loop connections. To optimize these relation embeddings and the other parameters simultaneously, a gradient scaling factor is proposed to constrain the embeddings to converge to suitable values. Besides, we theoretically demonstrate that our RE-GNNs have more expressive power than the meta-path based heterogeneous GNNs. Extensive experiments on the node classification tasks validate the effectiveness of our proposed method.

CCS CONCEPTS

• Computing methodologies → Neural networks; • Information systems → Social networks.

KEYWORDS

Heterogeneous graph, Graph neural networks

1 INTRODUCTION

Graph Neural Networks (GNNs) have shown great expressive power in graph representation learning [12, 15, 18, 32, 33]. They have been widely adopted in various downstream applications, such as social analysis [19, 26], biology prediction [8, 29], traffic prediction [11, 20], drug discovery [36], etc.

Unfortunately, the majorities of traditional GNNs are designed for homogeneous graphs, which disregard the variations in node type and edge type. The potential of GNN to handle the data with complex relations is not fully exploited. Currently, heterogeneous graph, a.k.a, heterogeneous information network, which possesses more than one type of nodes or edges, has attracted more attentions from researchers, due to its ability to represent data with more complicated relations.

Recently, several literatures have generalized homogeneous GNNs to handle the heterogeneous graphs. Existing heterogeneous GNNs can be classified into two categories based on their mechanisms for handling the heterogeneity.

The first type of approaches utilizes the composite relation, i.e., meta-path, to convert the heterogeneous graph to several meta-path based homogeneous graphs, as shown in Fig. 1(b). For example, in ACM datasets, the papers, which are not directly connected in the original heterogeneous graph yet connected with the same authors, are connected via the composite relation of paper-author-paper. These meta-path based heterogeneous GNNs [4, 9, 34, 38] utilize handcrafted or learned meta-paths as shortcuts to boost the efficiency of the message passing among the target types of nodes. However, they require certain prior knowledge to design the meta-paths [9, 34] or generate the meta-paths with excessive computations [4, 38]. Besides, the performance of these methods is highly correlated to the quality of the constructed meta-paths.

The other kind of approaches directly models the heterogeneity, especially the heterogeneous relations. Some methods handle the subgraph of each type of relations separately via separate models [14, 28, 39]. Since their complexities are correlated to the number of relation types, they are less efficient when there exists many...
Although [22] then proposes an improvement of GAT to process each type of nodes can pass messages according to various relations. Besides, excessive number of separate models usually leads to over-parameterization. Other methods generate the relation representations via specific modules and utilize them as a part of the neighbor messages [40], i.e., concatenate them with the node representations. Thus, in the aggregation step, node attributes and relation representations are aggregated orthogonally, i.e., the node representations are aggregated in a homogeneous manner.

Recently, [22] indicates that the homogeneous GNNs possess more potential to handle the heterogeneous graphs than previously reported [9, 34, 38]. It firstly reduces the complexity of the heterogeneous graph by converting its nodes and edges to a single type. Then, a specific homogeneous GNN, such as Graph Attention Network (GAT) [32], is applied and it can achieve comparable performance with the well-designed feature initialization strategies. Although [22] then proposes an improvement of GAT to process the heterogeneous graphs, it has not provided an explicit and intuitive mechanism for handling heterogeneity with the homogeneous GNNs. This motivates us to further explore the potentials of homogeneous GNNs in processing the heterogeneous graphs.

This paper aims to propose a simple yet efficient framework, to assign adequate abilities to the homogeneous GNNs for handling the heterogeneous graphs. The key of this framework is to effectively convert the entire to-be-processed heterogeneous graph to one homogeneous graph, i.e., to homogenize the various types of nodes and relations. Generally, the attributes in different types of nodes are extracted from different perspectives. To homogenize the graph, the features in different types of nodes are firstly projected into the same feature space, by assuming that different types of nodes are correlated implicitly. For example, in the heterogeneous academic network, all types of nodes, e.g., papers, authors, conferences and subjects, are actually correlated in a certain extent. Then, each type of nodes can pass messages according to various relations.

Considering the heterogeneity of relations, in the neighbourhood aggregation step, the messages from the nodes with different relations should not possess the same importance. Therefore, we must determine the appropriate importance for each relation in the neighbourhood aggregation step.

Specifically, we propose Relation Embedding based Graph Neural Networks (RE-GNNs), where only one embedding parameter is employed for each relation to model the aggregation importance. Then, the heterogeneous graph is converted to a weighted homogeneous graph by the proposed relation embeddings, as shown in Fig. 1(d). Similar to the proposed relation embeddings, node-type-specific self-loop embeddings are exploited to add the self-loop connections. Then, the weighted graph with self-loops can be directly processed by the traditional homogeneous GNNs [18, 32, 37]. Since the heterogeneity is embedded into the weighted homogeneous graph, the importance of a neighbor in the aggregation step is highly correlated with the type of their relation. Thus, in this simple framework, homogeneous GNNs can possess adequate abilities to handle the heterogeneous graphs.

To effectively model the heterogeneity, the learned weights of distinct relations are expected to be highly distinguishable. However, since the numerical values of relation embeddings are much larger than the other parameters, a straightforward simultaneous optimization of the relation embeddings and the other parameters cannot fully exploit the potential of relation embeddings. To tackle this numerical inconsistency, a gradient scaling factor is proposed to enlarge/suppress the updating modifications to the embeddings in each iteration, which constrains the relation embeddings to gradually converge to the expected values, thus generate effective weights of the relations.

We analyze the proposed RE-GNNs from two perspectives, i.e., spatial and meta-path based explanations. Note that although RE-GNNs are designed from the perspective of spatial aggregations, they also possess a meta-path based explanation, i.e., RE-GNNs can implicitly learn the composite relations from the heterogeneous graphs. Specifically, the standard GCN [18] with our relation embeddings, i.e., RE-GCN, is utilized as an example. By removing the nonlinearity and coarsening the weight matrices like SGC, we demonstrate that simplified RE-GCN can degenerate to the Graph Transformer Network (GTN) [38], a typical meta-path based heterogeneous GNN which automatically generates composite relations. Besides, we theoretically demonstrate that our RE-GCN possesses more expressive power than GTN, without explicitly generating the meta-paths.

Our contributions are as follows:

- We propose a simple yet efficient relation embeddings based framework to assign adequate ability to the homogeneous graph neural networks for handling heterogeneous graphs.
- We theoretically demonstrate that the proposed RE-GNNs possess more expressive power than typical meta-path based heterogeneous GNNs.
- To tackle the numerical inconsistency of the relation embeddings with other parameters, we propose a gradient scaling factor to effectively optimize the relation embeddings and other model parameters simultaneously.
2 RELATED WORK

2.1 Graph Neural Networks

Inspired by the traditional deep neural networks, Graph Neural Networks [27] are designed to handle the irregular graph data. The GNNs are usually designed from either the spectral or spatial perspectives. Spectral methods employ the graph spectral theory to define the graph convolution operation [1, 3, 18]. Spatial methods design the graph convolution operation directly on the graph, and aggregate the message from the spatial neighbors [2, 12, 32]. Recently, researchers [37] discovered that the former GNNs are actually less powerful than the traditional Weisfeiler-Lehman (WL) algorithm in the graph classification task. Then, various WL-GNNs [21, 23, 37] are proposed to enlarge the theoretical expressive power of the GNNs. Unfortunately, most of them are only designed to handle the homogeneous graphs.

2.2 GNNs for Heterogeneous Graphs

To process the heterogeneous graph, researchers generalize the traditional homogeneous GNNs to heterogeneous graphs. There exists two types of approaches to model the heterogeneity.

The first type of approaches is based on meta-path constructions, which is firstly introduced by HAN [34]. It converts a heterogeneous graph to multiple homogeneous graphs by various manually-designed meta-paths. For each type of meta-paths, the meta-path based neighbors are connected in the corresponding homogeneous graph. Then, the results of each meta-path based homogeneous graph are fused by attention. Based on the above mechanism, MAGNN [9] utilizes the intermediate nodes along the meta-path instead of only considering the meta-path based neighbors. These two methods require certain prior domain knowledge to design the meta-paths for each heterogeneous graph. GTN [38] generates meta-paths via a soft selection of edge types. Then an ensemble of GCNs is utilized to process the learned composite relations. Subsequently, the neural architecture search (NAS) technique [7] is utilized in DiffMG [4] to seek for a suitable meta-graph to model the more complicated composite relations.

The other type of approaches intends to model the heterogeneity directly via different kinds of nodes and relations. RGCN [28] models the relations in knowledge graphs by employing specialized parameter matrices, which separately constructs GNNs on each relation graph. HGT [14] further utilizes a transformer network for each relation to model the importance of each relation, according to the connected node attributes. RHIN [40] constructs an edge-centric coarsened line graph to generate the relation representations, and then transfers the content of the node and relation representations to the target nodes, in the message passing process. HGB [22] generalizes the GAT [32] by adding the edge type attention to the original pair-wise self-attention.

3 PRELIMINARIES

3.1 Heterogeneous Graph

A graph is defined as \( G = (<V,E,F,R,\phi,\psi>\) where \( V \) and \( E \) stand for the collections of nodes and edges, respectively. \( \phi : V \rightarrow F \) and \( \psi : E \rightarrow R \) are the node type and edge type mapping functions. \( F = \{ \phi(v) : \forall v \in V \} \) is the dynamic range of the node type projections and \( R = \{ \psi(e) : \forall e \in E \} \) is the dynamic range of the edge type projections. A heterogeneous graph [30] contains more than one type of nodes or edges, i.e., \( |F|+|R| \geq 2 \). \( A_t \) denotes the corresponding adjacency matrix of the edge type \( i \in R \). On the contrary, in homogeneous graphs, both the node and edge types only possess one valid value, i.e., \( |F| = |R| = 1 \).

3.2 Meta-path

Meta-path is widely adopted in the learning of heterogeneous graphs. A meta-path models a path passing through multiple relations (which can belong to different types of relations), e.g., \( v_1 \xrightarrow{e_1} v_2 \xrightarrow{e_2} \cdots \xrightarrow{e_{l-1}} v_{l+1} \), where \( v_i \in V \) and \( r_j \in R \). It describes every node pair \( v_1 \) and \( v_{l+1} \), which is connected by a composite relation. Note that \( R = r_1 \circ r_2 \circ \cdots \circ r_l \), where \( \circ \) denotes the composition operator on relations. \( v_1 \) and \( v_2 \) are the meta-path based neighbors, and the corresponding composite adjacency matrix is \( A_P = A_r A_r \cdots A_r \).

3.3 Graph Convolutional Network

Graph Convolutional Network (GCN) [18] has become the most popular GNN in the past few years. Given a heterogeneous graph \( G \), the graph convolution operation can be described as

\[
H^{l+1} = \sigma(\tilde{A}H^{l}W^{l}),
\]

where \( \tilde{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \) is a normalized adjacency matrix, \( \tilde{A} = A + I \) denotes the adjacency matrix with self-loops, and \( \tilde{D} \) represents the corresponding degree matrix. \( W^{l} \in \mathbb{R}^{n_k \times d_{out}} \) contains the learnable parameters. \( H^{l+1} \) is the output of the \( l \)-th layer and the input of the \( (l+1) \)-th layer, and \( H^{1} = X \). \( \sigma \) stands for the non-linear activation function, e.g., ReLU. For a directed graph (i.e., asymmetric adjacency matrix), \( \tilde{A} \) in Eq. (1) can be normalized by the inverse of the degree matrix, i.e., \( \tilde{D}^{-1} \), as \( \tilde{A} = \tilde{D}^{-1} \tilde{A} \).

3.4 Graph Transformer Network

Graph Transformer Network (GTN) [38] generates meta-paths in an end-to-end manner, which possesses the ability to search for the task-specific meta-paths from all the possible meta-paths. Since GTN is a typical meta-path based heterogeneous GNN and it will be compared to our approach in later sections, we perform a brief review here.

Firstly, GTN learns an \( l \)-length composite relation via a GT layer

\[
A_P = \left( \sum_{r_1 \in R} a_{r_1}^{(1)} A_{r_1} \right) \cdots \left( \sum_{r_l \in R} a_{r_l}^{(l)} A_{r_l} \right),
\]

where \( a_{r_l}^{(l)} = \text{softmax}_j(w_{r_lj}) \) is the soft weight for the edge type \( r_l \in R \) and \( w \) is the learnable parameters.

Then, the learned composite relation is fed into the standard GCN as

\[
Z^{(l)} = \sigma(\tilde{D}^{-1} \tilde{A}_P H^{(l)}W^{(l)}),
\]

where \( \tilde{A}_P^{(l)} = A_P^{(l)} + I \) is the composite relation adjacency matrix (with self-loops) and \( \tilde{D}_P \) denotes the corresponding degree matrix.
At last, the representations learned by multiple generated meta-paths are concatenated as
\[ H^{(l+1)} = ||_{l=1}^{C} Z_i^{(l)} \],
where || is the concatenation operator and C represents the number of generated meta-paths. This architecture can be viewed as an ensemble of GCNs on multiple generated meta-path relations.

4 METHODOLOGY
To assign adequate abilities to the homogeneous GNNs for handling the heterogeneous graphs, in this section, we propose a simple yet efficient framework, named Relation Embedding based Graph Neural Networks (RE-GNNs). Specifically, by assuming that all types of nodes are associated in a particular situation, the node features are projected via a node-type-specific linear transformation. Then, for the heterogeneous topology, one embedding parameter per relation is exploited to learn the importance of different types of relations and self-loop connections. Then, the heterogeneous graph can be converted to a weighted homogeneous graph, which can be handled by typical homogeneous GNNs. Note that since the linear transformation can be combined with the first GNN layer, this framework only introduces one parameter for each relation, compared to the original homogeneous GNNs. Besides, to ensure that the proposed relation embeddings can acquire effective information, a gradient scaling factor is proposed to adjust the updates in each optimizing iteration, which constrain the weights of the relations to converge to a proper value. The detailed illustration of our RE-GNNs is presented in Fig. 2.

4.1 Feature Projection
Usually, the features in different types of nodes are extracted via various schemes, while these nodes are always associated in certain conditions. Therefore, the features in each node are firstly projected via a type-specific linear transformation \([34, 38]\). The projected features are represented as
\[ \tilde{x}_i = x_i W\phi(i), \]
where \(x_i\) is the original features in node \(i\) and \(W\phi(i)\) is the learnable projection matrix for the node type \(\phi(i)\). Let \(\tilde{X} = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)\) be the matrix form of the features after transformation.

4.2 Relation Embeddings
After projecting the attributes of each type of nodes into the same feature space, nodes can exchange message with all their neighbors from different types of relations. In the aggregation step, the message from different types of edges should possess different importances. Then, one relation embedding value is utilized for each edge type, to learn its importance. The learned weighted adjacency matrix is formulated as
\[ A_H^{(l)} = \sum_{i \in R} \tau(e_i^{(l)}) A_i, \]
where \(e_i^{(l)} \in \mathbb{R}\) stands for the learnable relation embedding of edge type \(i \in R\) for the \(l\)-th layer, and \(\tau(\cdot)\) is a function to generate the aggregation importance from the embeddings. For simplicity, we set \(\tau\) as the LeakyReLU function to ensure the importance of different relations to be non-negative. In different layers, the relation embeddings can learn different importance weights for various relation types.

Similarly, we consider the self-loop connections, which connects the node with itself, as a special type of relations. Here, the same relation embeddings are employed to learn the importances of node-type-specific self-loop connections. The adjacency matrix with self-loops is formulated as
\[ \hat{A}_H^{(l)} = A_H^{(l)} + \sum_{j \in F} \tau(s_j^{(l)}) I_j, \]
where \(I_j\) is the diagonal matrix. If and only if \(\phi(i) = j\) and \((I_j)_{ii} = 1\),

Then, the learned weighted adjacency matrix can be utilized by the homogeneous GNN layers, i.e.,
\[ H^{(l+1)} = \text{GNNLayer}(A_H^{(l)}, H^{(l)}), \]
where \(H^{(0)} = \tilde{X}\) represents the projected node features and the GNNLayer can be the layers in any homogeneous GNN, such as GCN and GAT. If GCN is employed as the baseline model, the RE-GCN layers can then be represented by
\[ H^{(l+1)} = \sigma(\hat{A}_H^{(l)} H^{(l)} W(l)), \]
where \(\hat{A}_H^{(l)}\) is the normalized version of \(\hat{A}_H^{(l)}\). Note that the symmetric normalization version is \(\hat{A}_H = \hat{D}_H^{-\frac{1}{2}} \hat{A}_H \hat{D}_H^{-\frac{1}{2}}\), which is usually utilized in processing the undirected graphs. Meanwhile, the asymmetric normalization version is \(\hat{A}_H = \hat{D}_H^{-\frac{1}{2}} \hat{A}_H\), which is usually applied to the directed graphs. \(\hat{D}_H\) denotes the degree diagonal matrix corresponding to \(\hat{A}\).

4.3 Gradient Scaling
In the optimization process, the relation embeddings are firstly initialized as ones, where each type of relations has identical importance. With the network training, the weights of the relations are expected to diversify. Since the absolute value of embedding parameters are much larger than the other parameters, the learned embeddings are not differentiable enough. For example, a regular parameter \(w\) is initialized via the Xavier Uniform \([10]\) method by setting \(|w| < a\), where \(a\) is a small value correlated to the input and output dimensions and the relation embeddings are set to ones. Due to the above numerical inconsistency, it is difficult for the popular optimizers to simultaneously optimize the relation embeddings and model parameters. To tackle this problem, a gradient scaling factor is proposed to enlarge the gradients of the weights of the relations.

Specifically, a pre-defined scaling factor \(\lambda > 0\) is exploited on the original embeddings as
\[ a_i = \lambda e_i. \]
Then, the scaled weights are utilized to generate the adjacency matrix as
\[ \hat{A}_H = \sum_{i \in R} \tau(a_i) A_i. \]
To initialize the scaled weights as ones, each embedding parameter \(e_i\) is initially set to \(\frac{1}{\sqrt{2}}\). Then, the value of the scaling factor \(\lambda\) will affect the updating modification to \(a\). In general, if \(\lambda \geq 1\), it serves as a gradient enlarging machine and vice versa. A concise
5 ANALYSIS

In this section, we analyze the efficiency of our RE-GNNs from two distinct perspectives. Firstly, we provide a spatial analysis which intuitively illustrates the effectiveness of our framework. Then, we reveal that our RE-GNNs can be interpreted as the meta-path based heterogeneous GNNs. Specifically, when employing GCN as the backbone, our RE-GCN can degenerate to a meta-path based heterogeneous GNN in certain conditions. Besides, we also theoretically demonstrate that RE-GCN possesses more expressive power than the meta-path based heterogeneous GNN.

5.1 Spatial Analysis

Under the assumption that all types of nodes are implicitly correlated, RE-GNNs firstly project the raw features of different types in nodes into the same feature space. Then RE-GNNs learn a weighted relation adjacency matrix for the homogeneous GNNs, to model the importance of different relations. For example, for a paper node in the ACM network, different messages from different relations (e.g., Paper-Paper, Conference-Paper, Subject-Paper and node type specific self-loop (Paper)) are aggregated based on the corresponding relational importances. Similarly, for a conference, the message passed by Paper-Conference relation and self-loop (Conference) connection are aggregated based on the weights of the relations.

5.2 Meta-Path based Analysis

Besides of the explanation from the perspective of spatial aggregations, RE-GNNs also possess an implicit explanation, i.e., RE-GNNs can be interpreted as the meta-path based heterogeneous GNNs. For convenience, Graph Transformer Network (GTN) [38] is selected as the compared heterogeneous GNN. Our RE-GNNs framework also employ GCN [18] as the backbone, similar to GTN. Here, we reveal that our RE-GCN can degenerate to GTN in certain conditions. Besides, we also demonstrate that RE-GCN possesses more expressive power than GTN, theoretically.

5.2.1 Simplifying RE-GCN. As stated in Sec. 3.4, GTN can be regarded as an ensemble of GCNs on multiple generated composite relations. By excluding the ensemble trick in Eq. (4), we consider a simple case where GTN only learns a 2-length composite relation, as below.

\[
Z_p = \sigma \left( \tilde{A}_2 H^{(0)} W_p^{(0)} \right) \tag{13}
\]

where \( \tilde{A}_2 = \tilde{D}_2^{-1/2} \tilde{A}_2 \tilde{D}_2^{-1/2} \), \( A_2 = A_2 + I \) and \( \tilde{D}_2 \) represents the degree matrix of \( A_2 \). \( A_2 = (\sum_{i \in R} a_{1,i} A_1) (\sum_{j \in R} a_{2,j} A_2) \) is the learned 2-length meta-path. By stacking two RE-GCN layers, we can obtain

\[
Z_H = \sigma \left( \tilde{A}_H^{(1)} \sigma \left( \tilde{A}_H^{(0)} H^{(0)} W_H^{(0)} \right) W_H^{(1)} \right) \tag{14}
\]
Given an MLP layer \( f : \mathcal{H}_\text{in} \to \mathcal{H}_\text{out} \) bounded by \( k \), there exists a composite of two MLP layers \( (f_1 \circ f_2) \) which equals to \( f \), where \( f_1 \) is also bounded by \( k \) and \( f_2 \) is bounded by \( \max\{1, 2k\} \).

Since a particular solution of the layer \( f_2 \) is \( W_2 \) being an identity matrix when \( b_2 \) is bounded by \( 2k \), the layer \( f_2 \) is bounded by \( \max\{1, 2k\} \) in Lemma 3. By considering that the bias plays a complementary role to parameter \( W \), we do not separately restrict the bound of \( b \), in practice. Thus, it is acceptable that the bias of \( f_2 \), i.e., \( b_2 \), has a bit larger bound.

According to Lemma 3, we can conclude that a composite of two MLP layers possesses no less expressive power than one MLP layer.

Now, let us consider the scenario of GNN. Similarly, for a GCN layer, we still utilize \( k = \max\{k_{in}, k_{w}, k_b\} \) as its bound, where \( k_{in}, k_{w}, \) and \( k_b \) are the bounds of the input, parameter and bias, respectively. In practice, GCN layer usually utilizes a bias vector. For example, the GTN layer in Eq. (13) can be rewritten as

\[
Z_P = \sigma \left( \hat{A}_H \hat{W}_P + B_P \right),
\]

where \( B = \begin{pmatrix} b \\ b \end{pmatrix} \) is the broadcast of a bias vector \( b \) (stacking multiple rows with a row vector \( b \)). The GTN layer actually adds a bias vector to each sample. Similarly, a composite of two RE-GCN layers in Eq. (14) can be rewritten as

\[
Z_H = \sigma \left( \hat{A}_H^{(1)} \hat{W}_H^{(0)} + B_H^{(0)} \right) \hat{W}_H^{(1)} + B_H^{(1)}.
\]

Lemma 4.

If a vector set \( \mathcal{H} \) is bounded by \( k \), then for each \( \mathcal{H} \in \mathcal{H} \) bounded by \( k \), there exists a composite of two MLP layers, where the first layer is bounded by \( k \) and the second layer is bounded by \( \max\{1, 2\} \), which is equivalent to \( f_{\text{GCN}} \).

Corollary 5.

If \( f_{\text{GTN}} : \mathcal{G} \to \mathbb{R}^{N \times C} \) is a 2-lengthed GTN layer, which is bounded by \( \xi \), there exists a composite of two RE-GCN layers, where the first layer is bounded by \( \xi \) and the second layer is bounded by \( \max\{1, 2\} \), which is equivalent to \( f_{\text{GTN}} \).

Corollary 5 theoretically reveals that a composite of two RE-GCN layers possesses no less expressive powers than a 2-lengthed GTN layer. Then, we can extend it to the general case.

Theorem 6.

Let \( G \in \mathcal{G} \) be a heterogeneous graph, where the node features \( X \in \mathcal{X} \) are normalized, i.e., \( \forall X \in \mathcal{X}, |X| < \xi \). If a non-ensembled GTN, \( m : \mathcal{G} \to \mathbb{R}^{N \times C} \), maps the nodes in \( G \) to any node embeddings \( Z \in \mathbb{R}^{N \times C} \), there exists a RE-GCN which is equivalent to GTN.

Theorem 6 further demonstrates that RE-GCN possesses no less expressive power than the GTN. Then, Theorem 7 is obtained as follows.

Theorem 7.

Let \( G \in \mathcal{G} \) be a heterogeneous graph, where the node features \( X \in \mathcal{X} \) are normalized, i.e., \( \forall X \in \mathcal{X}, |X| < \xi \). There exists a RE-GCN, \( r : \mathcal{G} \to \mathbb{R}^{N \times C} \), which can map \( G \) to the node embeddings \( Z \in \mathbb{R}^{N \times C} \) that GTN cannot map \( G \) to.

Theorems 6 and 7 jointly prove that our RE-GCN possesses more expressive power than the GTN. According to the above analysis, we can conclude that RE-GCN can be interpreted as an implicit...
Table 1: Statistics of the datasets.

| Datasets | Nodes | Edges |
|----------|-------|-------|
| DLBL | # author(A):4,057 # paper(P):14,328 # term(T):7,723 # venue(V):20 | # A-P:19,645 | # P-T:85,810 | # P-V:14,328 |
| ACM | # paper(P):4,019 # author(A):7,167 # subject(S):50 | # P-P:9,615 | # P-A:13,407 | # P-S:4,019 |
| IMDB | # movie(M): 4,278 # director(D): 2,081 # actor(A): 5,257 | # M-D:4,278 | # M-A:12,828 |

meta-path based heterogeneous GNN, which indicates that it may be unnecessary to pay too much attentions to design or generate the meta-path explicitly. By simply stacking multiple RE-GNN layers, the heterogeneity can also be efficiently handled.

6 EVALUATIONS

6.1 Datasets

Three widely utilized heterogeneous datasets, i.e., two heterogeneous academic networks, DLBL and ACM, and one heterogeneous movie network IMDB, are employed to demonstrate the effectiveness of our RE-GNNs. Their details are shown in Table 1.

In DLBL, four types of nodes, i.e., papers, authors, venue, and terms, are constructed. There are three types of relations, i.e., A-P, P-T, and P-V, which possess directions for our RE-GNNs. According to the conferences they submitted, authors are categorized into four research areas, i.e., Database, Data Mining, Artificial Intelligence and Information Retrieval. Each paper is described by a bag-of-words (BOW) representation of its keywords, and each author is described by the BOW embeddings of its published papers. Each term is represented by the Glove word vectors [24], and the attributes of venues are formed as one-hot vectors.

In ACM, three types of nodes, i.e., papers, authors, and subjects, and three types of relations are constructed. According to the conference they published, papers are divided into three categories, i.e., Database, Wireless Communication, and Data Mining. Each paper is described by the BOW representation of keywords. The attributes of authors and subjects are formed as one-hot vectors.

In IMDB, three types of nodes, i.e., movies, directors, and actors, and two types of relations are formed. According to their genre, the movies are classified into three categories, i.e., Action, Comedy, and Drama. Movies are described by the BOW embeddings of their plots. The attributes of actors and directors are represented by the mean of the attributes of their related movies.

6.2 Baselines

We compare our method with various kinds of popular GNN methods, including homogeneous GNNs (GCN [18] and GAT [32]), heterogeneous graph embedding methods (Metapath2vec [5]), meta-path based heterogeneous GNNs (HAN [34], GTN [38] and MAGNN with Attributes Completion [9, 16]), relation based heterogeneous GNNs (RGGCN [28], HetGNN [39] and HGB [22]). Note that the homogeneous GNNs, i.e., GCN and GAT, are constructed on the homogeneous graph, where the heterogeneity is directly eliminated. Here, GCN-M and GAT-M respectively represent GCN and GAT constructed on several meta-path based homogeneous graphs, and the best scores are reported.

6.3 Implementation Details

For the homogeneous GNNs, four-layered GCN and GAT, with 64 hidden neurons and 4 heads (only for GAT), are employed as our baselines. Note that, we employ GAT with 128 hidden neurons and 1 heads on IMDB, which are obtained via hyperparameter search. For the corresponding RE-GNNs, the gradient scaling factor $\lambda$ is set to 100. In the training process, both the homogeneous GNNs and RE-GNNs are trained for a maximum of 200 epochs with an early stopping condition at 50 epochs. The cross-entropy loss is utilized as the loss function and Adam [17] optimizer is employed with the learning rate of 0.001, the weight decay rate of 0.001 for ACM and DLBL, and 0.005 for IMDB. The dropout layers are utilized for the input of each GNN layer in the training process with a dropout rate of 0.6.

6.4 Main Results

The node classification results of three heterogeneous datasets, i.e., DLBL, ACM, and IMDB, are presented in Tab. 2. For the baselines, both the meta-path based and relation based heterogeneous GNNs achieve outstanding performances. For example, MAGNN-AC [16] achieves 94.61, 93.87, and 60.98 Micro-F1 scores on these datasets. Unfortunately, these methods design specific complicated modules to process the heterogeneous graphs. On the contrary, our RE-GNNs, which only utilize one-dimensional embeddings, can
allow the homogeneous GNNs to effectively process the heterogeneous graphs. For example, in DBLP, RE-GCN performs much better than the original GCN, which has 7.84 and 6.51 improvements in Macro-F1 and Micro-F1 scores, respectively. A similar trend happens on GAT and RE-GAT. Meanwhile, for the ACM dataset, although our RE-GCN still achieves the best performance, the performance gain is relatively small, e.g., 0.28 in the Macro-F1 score. The reason is that the weights of the relations with the current initialization may be already close to the optimal weights in some particular cases, though our RE-GNNs aim to search for the optimal weights of the relations. Under such circumstances, the performances of homogeneous GNNs may be close to or even approximately equal to the corresponding RE-GNNs. In general, our RE-GCN achieves better performance than the other well-designed heterogeneous GNNs. According to these results, our RE-GNNs can effectively assign adequate abilities to the homogeneous GNNs to handle heterogeneous graphs.

### 6.5 Visualizations

For visual comparisons, we utilize T-SNE [31] to project the learned embeddings of the authors in DBLP datasets into a 2-dimensional space. As shown in Fig. 3(a), GCN, which is a standard homogeneous GNN, does not perform well on the heterogeneous graphs for the node classification task. Authors with identical research interest tend to be decentralized, and authors belonging to different categories tend to be mixed. A similar situation happens to GAT, as shown in Fig. 3(c). On the contrary, after introducing the relation embeddings to model the heterogeneity, RE-GNNs can effectively learn suitable embeddings. The embeddings learned by RE-GCN and RE-GAT are given in Figs. 3(b) and 3(d). The author with identical research interest are more compact, and authors with different research interests are more distinguishable. These results further validate the effectiveness of our relation embedding framework.

### 6.6 Ablation Study

Here, we verify the effectiveness of the edge-type relation embeddings and self-loop relation embeddings. Two variants of our RE-GNNs are given: 1) GNN-S: homogeneous GNNs with the node type specific self-loop embeddings; 2) GNN-E: homogeneous GNNs with the edge type relation embeddings and general self-loop embeddings.

As shown in Table 3, the edge type relation embeddings are very effective, e.g., the F1 scores of GNN-E are approximately 6.0 points higher than GNNs. Meanwhile, the node type specific self-loop embeddings give relatively small improvements, because the normalizations in the aggregation step of GNNs will normalize
the message from different types of relations and self-loops jointly. Then, the same normalized result can be obtained by subjecting the weights of the self-loop connections as one. Although GNN-E can implicitly learn the self-loop weights, it is not designed to perform this function. Therefore, we constrain RE-GNNs to explicitly learn these weights, whose effectiveness has been proved by the experimental results.

6.7 Impacts of Gradient Scaling Factor

As stated in Sec. 4.3, the update of the weights of the relations in each optimization step can be scaled via the gradient scaling factor $\lambda$, theoretically. By using the Adam optimizer, the update of the weights of the relations changes $\lambda$ times. Here, we experimentally verify the effectiveness of the proposed gradient scaling factor. Fig. 4 presents the results of the first layer of RE-GCN learned on the DBLP dataset. When $\lambda$ is set to 0.001, the update of the weights of the relations is negligible, so RE-GCN becomes an approximation of GCN with the fixed weights of the relations being one. When $\lambda$ is set to 1.0, RE-GCN possesses the original relation embeddings in Eq. (6). As shown in Fig. 4(b), the final weights of the relations are similar. When we set a proper scaling factor, e.g., $\lambda = 100$, RE-GCN can learn distinguishable weights of the relations. As can be observed from the learned weights of the relations, the first layer of RE-GCN believes that the paper attributes are more critical, and thus the importance of P-* (i.e., P-A, P-T, P-C, and P) relations are much larger than the others. When $\lambda$ is too large, e.g., 1000, the weights of the relations are easy to fall into polarization, i.e., part of the relations are dominating the other relations. In summary, the gradient scaling factor can help the RE-GNNs learn the proper relation embeddings and process the heterogeneous graphs.

7 CONCLUSION

This paper proposes a simple yet efficient framework, named Relation Embedding based Graph Neural Networks (RE-GNNs), to assign adequate ability to the homogeneous graph neural networks for handling the heterogeneous graphs. Specifically, we utilize only one parameter per relation to model the importance of the distinct types of relations and node-type-specific self-loop connections. To optimize the relation embeddings and the model parameters simultaneously, a gradient scaling factor is proposed to enable the embeddings to converge to appropriate values. Besides, we theoretically and experimentally demonstrate that our RE-GNNs have more expressive power than the meta-path based heterogeneous GNNs. Extensive experiments have validated the effectiveness of the proposed RE-GNNs.

REFERENCES

[1] Ioan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In ICLR.
[2] Jie Chen, Tingfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In ICLR.
[3] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In NIPS. 253–261.
[4] Yuhui Ding, Quanming Yao, Huan Zhao, and Tong Zhang. 2021. DiffMG: Differentiable Meta Graph Search for Heterogeneous Graph Neural Networks. In ACM SIGKDD. 279–288.
[5] Yuixiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In ACM SIGKDD. 135–144.
[6] John Duch, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. JMLR 12, 7 (2011).
[7] Thomas Eisken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. JMLR 20, 1 (2019), 1997–2037.
[8] Alex Fout, Jonathon Byrd, Basil Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In NIPS. 6533–6542.
[9] Xinyu Fu, Jian Zhang, Zijiao Meng, and Irwin King. 2020. Magm: Metapath aggregated graph neural network for heterogeneous graph embedding. In WWW. 2331–2341.
[10] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In AISTATS. 249–256.
[11] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In AAAI. 922–929.
[12] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Represen-
tation Learning on Large Graphs. In NIPS. 1024–1034.
[13] Bingsheng He and Xiaoming Yuan. 2012. On the O(1/n) convergence rate of the Douglas–Rachford alternating direction method. SIAM J. Numer. Anal. 50, 2 (2012), 700–709.
[14] Zi Min Hu, Yuixiao Dong, Kuanwan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In WWW. 2704–2710.
[15] Bo Jiang, Ziyang Zhang, Doudou Liu, Jin Tang, and Bin Luo. 2019. Semi-Supervised Learning With Graph Learning-Convolutional Networks. In IEEE CVPR. 11313–11320.
[16] Di Jin, Cuiying Hua, Chunlong Ding, and Liang Yang. 2021. Heterogeneous Graph Neural Network via Attribute Completion. In WWW. 391–400.
[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Opti-
mization. In ICLR.
[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR.
[19] Chang Li and Dan Goldwasser. 2019. Encoding social information with graph convolutional networks for political perspective detection in news media. In ACL. 2594–2604.
[20] Jia Li, Zhihao Han, Hong Cheng, Jiao Su, Pengyun Wang, Jianfeng Zhang, and Lujia Pan. 2019. Predicting path failure in time-evolving graphs. In ACM SIGKDD. 1279–1289.
[21] Andreas Loukas. 2019. What graph neural networks cannot learn: depth vs width. ICLR (2019).
[22] Qingsong Lv, Ming Ding, Qiang Liu, Yuixiang Chen, Wenzheng Feng, Simon H, Chang Zhout, Jianguo Jiang, Yuixiao Dong, and Jie Tang. 2021. Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In ACM SIGKDD. 1150–1160.
[23] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weissfeiler and leman go neural: Higher-order graph neural networks. In AAAI. 4602–4609.
[24] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In EMNLP. 1532–1543.
[25] Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. Neural networks 12, 1 (1999), 145–151.
[26] Jiezhong Qiu, Jian Tang, Hao Ma, Yuixiao Dong, Kuanwan Wang, and Jie Tang. 2018. Deepin: Social influence prediction with deep learning. In ACM SIGKDD. 2110–2119.
[27] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. IEEE Transactions on Neural Networks 20, 1 (2008), 61–80.
A THEORETICAL IMPACTS OF GRADIENT SCALING FACTOR

Here, we discuss the theoretical impacts of the proposed gradients scaling factor, i.e., $\alpha_t = \lambda \epsilon_t$, where the relation embedding $\epsilon_t$ is scaled by a factor $\lambda > 0$. The gradient of object function $L$ to the weight of each relation $\alpha_t$ is $\frac{\partial L}{\partial \alpha_t}$. Then the gradient to each relation embedding $\epsilon_t$ is $\lambda \frac{\partial L}{\partial \epsilon_t}$. In the original case, where the relation embedding is utilized directly as the weight of the relation ($\alpha_t = \epsilon_t$), the gradient to $\epsilon_t$ is denoted as $g_t$. By employing the gradient scaling factor, the scaled gradient to $\epsilon_t$ is denoted as $g_t' = \lambda g_t$.

Now, we consider the updating modification for each iteration in the optimization process. For the gradient descent based optimizers, each relation embedding is updated via $\epsilon_t^{\text{next}} = \epsilon_t - \Delta \epsilon_t$, where $\Delta \epsilon_t = \kappa (g_t)$ is correlated to the gradient $g_t$. With the scaling factor $\lambda$, the updating modification becomes $\kappa (\lambda g_t)$.

We argue that $\kappa (\lambda g_t) = \lambda \kappa (g_t)$, for the common gradient optimizers such as SGD, Momentum and Nesterov Momentum. For the adaptive optimizers like Adagrad and Adam, $\kappa (\lambda g_t) = \kappa (g_t)$. Here, we present the detailed proofs of SGD and Adam optimizers as examples. The proofs of other commonly utilized optimizers can be generalized by these proofs.

Stochastic Gradient Descent (SGD). For a Stochastic Gradient Descent (SGD) optimizer, a parameter $\theta$ is updated by

$$\theta_t = \theta_{t-1} - \eta g_t,$$

where $\eta$ stands for the learning rate and $g_t = \frac{\partial L}{\partial \theta_{t-1}}$ represents the gradient of a batch of the input data. Thus, the updating modification function is $\kappa (g) = -\eta g$. Then, $\kappa (g') = \kappa (\lambda g) = -\eta \lambda g = \lambda (-\eta g) = \lambda \kappa (g)$.

Adaptive Moment Estimation (Adam). Adaptive Moment Estimation (Adam) is a typical gradient descent method which computes an adaptive learning rate for each parameter. In each iteration $t$, it firstly computes the exponential averages of the gradient $m_t$ and the squared gradient $v_t$ respectively as

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t,$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2,$$

where $\beta_1$ and $\beta_2$ are the two pre-defined hyperparameters. Both $m_t$ and $v_t$ are initialized as vectors of zeros. Then, the bias elimination is utilized to obtain

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t},$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}.$$

At last, the parameter is updated accordingly

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where $\epsilon > 0$ is an extremely small number which can be neglected when $v_t \neq 0$. Then, the updating modification function is $\kappa (g_t) = -\eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$. In the following proof, induction is employed to prove $\kappa (\lambda g_t) = \kappa (g_t)$.

Proof. Initially, we have $m_0 = 0$ and $v_0 = 0$. Let $g'_t = \lambda g_t$. Then, $\kappa (\lambda g_t) = \kappa (g')$.

In the first iteration, since $m_0 = v_0 = 0$, it is easy to validate $m'_1 = \lambda m_1, v'_1 = \lambda^2 v_1, \hat{m}'_1 = \lambda \hat{m}_1$ and $\hat{v}'_1 = \lambda^2 \hat{v}_1$. Then, the updating
Then, when Eq. (14) can be rewritten as
$$B PROOF OF LEMMA 3$

Proof. Let $$\Delta m_{i-1}$$ be the $$i$$-th row of $$\Delta m$$, which is only utilized to prevent a zero denominator in practice, we ignore its effect here. Assuming that in the $$(t - 1)$$-th iteration, we have obtained $$m_{t-1}^* = \lambda m_{t-1}$$ and $$v_{t-1}^* = \lambda^2 v_{t-1}$$ and $$\kappa(g_{t-1}) = \kappa(g_t)$$. Then, in the $t$-th iteration, we can obtain

$$m_t^* = \beta_1 m_{t-1} + (1 - \beta_1) v_t^*$$

$$= \beta_1 \lambda m_{t-1} + (1 - \beta_1) \lambda v_t$$

$$= \lambda (\beta_1 m_{t-1} + (1 - \beta_1) v_t)$$

$$= \lambda m_t.$$  \hspace{1cm} (S7)

Similarly, we can obtain

$$u_t^* = \beta_2 u_{t-1} + (1 - \beta_2) v_t^*$$

$$= \beta_2 \lambda^2 u_{t-1} + (1 - \beta_2) \lambda^2 v_t^*$$

$$= \lambda^2 (\beta_2 u_{t-1} + (1 - \beta_2) v_t^*),$$

Then, for the bias eliminated $$\hat{m}_t^*$$ and $$\hat{v}_t^*$$, we can compute $$\hat{m}_t^* = \lambda \hat{m}_t$$ and $$\hat{v}_t^* = \lambda^2 \hat{v}_t$$ respectively. At last, the updating modification is

$$\kappa(g_{t+1}) = \frac{\hat{m}_t^*}{\sqrt{\hat{v}_t^*}} = \kappa(g_t).$$ \hspace{1cm} \Box

Then, we consider the modification for the weight of each relation $$\Delta a_i$$. Since, $$a_i$$ is not a parameter, its modification is correlated to $$\lambda a_i$$. In the original case, $$\lambda a_i = \Delta a_i = \kappa(g_t)$$. By employing the gradient scaling factor, the modification of the weight of each relation is $$\Delta a' = \hat{\Delta} a_i = \lambda \kappa(g_t)$$. As stated above, for the common gradient optimizers, such as SGD, Momentum and Nesterov, $$\Delta a' = \lambda^2 \Delta a$$. For the adaptive optimizers like Adagrad and Adam, $$\Delta a' = \lambda \Delta a$$. \hspace{1cm} \Box

D PROOF OF COROLLARY 5

Proof. For Eq. (14), $$X$$ is the collection of $$x$$. Each row of $$X$$, which is denoted as $$x_t$$, is bounded by $$t, \overline{a} = \overline{D}^{-1} \overline{A}$$ is a non-negative matrix, where the summation of each row equals to $$1$$, i.e., $$\sum_j a_{ij} = 1$$ and $$\overline{a} = \sum_j a_{ij} = 0$$. Let $$H = \overline{A} X$$. According to Lemma 4, the bound of each row $$h_i$$ is $$\epsilon$$.

Then, according to Lemma 3, we can conclude that there exists a $$b'$$ with a bound of $$\frac{\epsilon + k}{2}$$, $$h Wi + b' > 0$$, i.e., $$h Wi + b' > 0$$. Thus, Eq. (14) can be rewritten as

$$Z_H = \sigma (A_H^{(1)} + A_H^{(0)} X W_H^{(0)} W_H^{(1)} + A_H^{(1)} B_H^{(0)} W_H^{(1)} + B_H^{(1)}).$$  \hspace{1cm} (S17)

Since $$A_H^{(1)}$$ is a matrix, which is normalized in each row, and $$B_H^{(0)}$$ is a column equivalent matrix, we can obtain

$$\hat{A}_H^{(1)} B_H^{(0)} = B_H^{(0)}.$$  \hspace{1cm} (S18)

Besides,

$$\hat{A}_H^{(1)} A_H^{(0)} = \hat{\hat{D}}(1) \hat{\hat{A}}_H^{(1)} \hat{\hat{A}}_H^{(0)} = \hat{\hat{D}}(1) \hat{\hat{D}}(0) \hat{\hat{A}}_H^{(1)} \hat{\hat{A}}_H^{(0)} = \hat{A}_H^{(1)} \hat{A}_H^{(0)} = \hat{A}_H^{(1)} H.$$

$$\hat{A}_H^{(1)} H = \hat{A}_H^{(1)} H = \hat{A}_H^{(1)} H = \hat{A}_H^{(1)} H = \hat{A}_H^{(1)} H.$$  \hspace{1cm} (S19)

With Eqs. (S18) and (S19), Eq. (S17) can be reformed to

$$Z_H = \sigma \left( \hat{A}_H^{(1)} X W_H^{(0)} W_H^{(1)} + B_H^{(0)} W_H^{(1)} + B_H^{(1)} \right).$$  \hspace{1cm} (S20)

Similar to the proof of Lemma 3, we can easily construct a solution that

$$W_H^{(0)} = W_p, W_H^{(1)} = I, B_H^{(1)} = B_p - B_H^{(0)}.$$ \hspace{1cm} \Box

E PROOF OF THEOREM 6

Proof. Theorem 1 can be proved in two steps.

- **T1(1).** For any $$L$$-lengthed one-layered GTN, there exists an $$L$$-layered RE-GCN which is equivalent to it.

- **T1(2).** For any $$L$$-lengthed $$K$$-layered GTN, there exists an $$(L K)$$-layered RE-GCN which is equivalent to it, where $$K > 1$$.

According to Corollary 5, for an $$L$$-lengthed GTN layer, we can also obtain a stack of $$L$$ RE-GCN layers which is equivalent to it. This equivalence can be achieved via removing the ReLU function and choosing a proper bias vector $$b^{(L)}$$ in each layer (except the
last layer). Note that the bias vector \( b^{(l)} \) is bounded by \( \frac{k^{(l)} + k^{(l)}}{2} \). \( k^{(l)}_{in} \) and \( k^{(l)}_{w} \) are the bounds of the input features and parameters in the \( L \)-th layer, respectively. Therefore, T1(i) is proved.

For T1(2), we can employ a composite of \( L \) RE-GCN layers which is equivalent to each \( L \)-lengthed GTN layer. Then, we can stack these \((KL)\) RE-GCN layers, which is equivalent to the \( L \)-lengthed \( K \)-layered GTN. Therefore, T1(2) is proved.

\( \square \)

F PROOF OF THEOREM 7
Proof. Theorem 2 can be proved in two steps.
• T2(1). There exists a 2-layered RE-GCN, which cannot be represented by any \( L \)-lengthed one-layered GTN.
• T2(2). There exists a 2-layered RE-GCN, which cannot be represented by any \( L \)-lengthed \( K \)-layered GTN, where \( K > 1 \).

Consider a graph with only one node and a self-loop connection. Then, a GCN layer is degenerated to an MLP layer. Since the composite of two MLP layers can be a non-linear mapping while one MLP layer can only be a linear mapping (by ignoring the last ReLU function), there exists a composite of two MLP layers that one MLP layer cannot be equivalent to. Therefore, T2(1) is proved.

The adjacency matrix enables the nodes (samples) to exchange messages with others. Then, the effects of neighbourhood aggregation with \( A \) cannot be replaced by the effects of weight projection with \( W \). For a \( K \)-layered GTN \((K > 1)\) with an arbitrary learned adjacency matrix \( A_p \), it should satisfy that \( A^{k_1}_H, A^{k_2}_H = A^{k_1 + k_2}_H \), where \( k_1 + k_2 = K \). However, this cannot be satisfied at all the time for any possible \( A^{k_1}_H \) and \( A^{k_2}_H \). For example, if \( A^{k_1}_H \) is a non-singular matrix, while \( A^{k_2}_H \) is a singular matrix, no proper \( A_p \) can be obtained. The reason is that \( A^p \) is a non-singular matrix, if and only if the matrix \( A \) is non-singular. Therefore, T2(1) is proved.

\( \square \)