iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** I **Month of publication:** January 2023

DOI: https://doi.org/10.22214/ijraset.2023.48485

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue I Jan 2023- Available at www.ijraset.com

An Approch for Representation of Node Using Graph Transformer Networks

Dev Vaghani
Department of Information Technology, G. H. Patel College of Engineering and Technology, Anand-388120, India

Abstract: ***In representation learning on graphs, graph neural networks (GNNs) have been widely employed and have attained cutting-edge performance in tasks like node categorization and link prediction. However, the majority of GNNs now in use are made to learn node representations on homogenous and fixed graphs. The limits are particularly significant when learning representations on a network that has been incorrectly described or one that is heterogeneous, or made up of different kinds of nodes and edges. This study proposes Graph Transformer Networks (GTNs), which may generate new network structures by finding valuable connections between disconnected nodes in the original graph and learning efficient node representation on the new graphs end-to-end. A basic layer of GTNs called the Graph Transformer layer learns a soft selection of edge types and composite relations to produce meaningful multi-hop connections known as meta-paths. This research demonstrates that GTNs can learn new graph structures from data and tasks without any prior domain expertise and that they can then use convolution on the new graphs to provide effective node representation. GTNs outperformed state-of-the-art approaches that need pre-defined meta-paths from domain knowledge in all three benchmark node classification tasks without the use of domain-specific graph pre-processing.***

I. INTRODUCTION

Deep networks can analyze structured inputs like molecules or social networks thanks to graph neural networks (GNNs). GNNs learn mappings from the structure and characteristics in their surroundings to calculate representations at graph nodes and/or edges. This local-neighborhood aggregation makes use of the relational inductive bias represented by the connectedness of the network [1]. By stacking layers, GNNs may gather data from outside of local neighborhoods similarly to convolutional neural networks (CNNs), hence expanding the GNN receptive field.

Since the creation of Transformers [2], which are now the top-performing neural network designs for handling long-term sequential datasets like sentences in NLP, there has been a huge amount of progress made in the field of natural language processing (NLP). This is accomplished through the employment of the attention mechanism [3], in which each word in a phrase pays attention to the words around it and integrates the information it receives to produce abstract feature representations. This technique of learning word feature representations by merging feature information from other words in a sentence may instead be considered as an instance of a GNN applied on a fully connected graph of words [5] from a message-passing paradigm [4] in graph neural networks (GNNs) viewpoint.

Graph Neural Networks (GNNs) have gained significant traction in recent years for a variety of graph-related tasks, including graph classification [22, 23, 24], link prediction [25, 26, 27], and node classification [28, 29, 30]. State-of-the-art performance in a range of graph datasets, including social networks [31,29,32], citation networks [33,30], the functional structure of brains [34], and recommender systems [35,36,37], has been demonstrated to be possible using the representation learned by GNNs. GNNs make use of the underlying graph structure to either execute convolution in the spectrum domain using the Fourier basis of a particular graph, i.e., the eigenfunctions of the Laplacian operator [38,3], or to operate convolution directly on graphs by passing node information [4,29] to neighbors.

However, a drawback of the majority of GNNs is that they presumptively function on fixed, homogenous graph structures. Since the fixed graph structure determines the previously stated graph convolutions, a noisy graph with missing or false connections yields an inefficient convolution with incorrect neighbors on the graph. Additionally, creating a graph to run GNNs is not always simple in some applications. For instance, a citation network is referred to as a heterogeneous graph because it has several node types (such as authors, papers, and conferences) and edges that are determined by the relationships between them (such as author-paper, and paper-conference). Neglecting the node/edge types and treating them as they are in a homogenous network is a naive approach (a standard graph with one type of node and edges).

This is not ideal because models cannot make use of the type of information. A more recent solution is to manually create meta-pathways, or paths joined by heterogeneous edges, and convert a heterogeneous network into a homogeneous graph that is defined by the meta-paths. The altered homogeneous graphs can then be used as input for standard GNNs [40, 41]. This strategy involves two stages and calls for individually created meta-paths for every issue. The selection of these meta-paths can have a big impact on how accurate downstream analysis is.

Modern performance on numerous NLP applications has been achieved using models based on transformers [6,7,8]. However, graph neural networks (GNNs) have shown to be the most successful neural network designs on graph datasets and have made substantial progress in a variety of fields, including physics [12,13], social sciences [11], and knowledge graphs [9,10]. In particular, GNNs take advantage of the flexible graph structure that is provided to learn the feature representations for the nodes and edges, and then they apply the learned representations to subsequent tasks.

For many computer vision tasks, such as object identification [14], picture classification [15], segmentation [16], facial recognition [17], and captioning [18], convolutional neural networks (CNNs) have become the de facto standard. The monkey visual system served as the inspiration for the inductive bias in CNNs, and the layer activations of these networks have been utilized to explain neural activations inside them [19]. Understanding the representations and techniques picked up by CNNs trained on well-known datasets like ImageNet has received a lot of attention recently [20,21]. Behavioral analysis analyzing model categories to learn more about the underlying representations—takes the form of much of this.

II. LITERATURE REVIEW

For a variety of purposes, several classes of GNNs have been created recently. They are divided into spectral [42] and non-spectral techniques [31] approaches. [42] suggested a method for performing spectral convolution in the spectrum domain by exploiting the Fourier basis of a certain graph, which is based on spectral graph theory. [33] convolution of the spectral graph with simplified GNNs of the first order. The definition of convolution operations, on the other hand, is done directly on the graph using spatially close neighbors in non-spectral techniques. Examples include [30]'s use of various weight matrices for nodes with various degrees and [29]'s learnable aggregator functions, which condense neighboring nodes' information for graph representation learning.

For many years, node categorization has been investigated. Traditionally, hand-crafted features like graph kernels, basic graph statistics, and engineering characteristics from a local neighbor structure have been utilized. These features lack flexibility and have subpar performance. Recently, node representation learning approaches using random walks on graphs have been suggested in Deep Walk [44], LINE [47], and node2vec [48], and have improved performance in part because of deep learning model tricks (such as skip-gram). All of these techniques, however, exclusively use the graph structure to learn node representation. The representations are not task-specifically optimized. GNNs learn a potent representation for the tasks and data they are given since CNNs have been so successful in representation learning. Generalized convolution based on spectral convolution [49, 50], attention mechanism on neighbors [51, 30], subsampling [43, 31], and inductive representation for a big network [29] have all been researched to enhance performance or scalability. Although these techniques produce excellent results, they are all constrained to work with homogeneous graphs.

But a lot of real-world issues frequently can't be modeled by a single homogenous graph. The graphs are delivered as heterogeneous graphs with a range of node and edge types. A two-stage method is one straightforward answer because most GNNs are made to work with a single homogenous graph. It pre-processes the heterogeneous graph into a homogeneous graph using meta-paths, which are the composite relations of various edge types and then learns representation. While HAN [40] learns graph representation learning by converting a heterogeneous graph into a homogeneous graph formed by meta-paths, the metapath2vec [51] learns graph representations by employing meta-path-based random walks. These methods, however, rely on manually chosen meta-paths made by subject matter experts, which means they might not be able to capture all significant linkages for each issue. Furthermore, the choice of meta-paths can have a big impact on performance. This Network Transformer network, in contrast to previous methods, can operate on a heterogeneous graph and transform it for tasks while learning end-to-end node representation on the altered graphs.

The over-smoothing issue is addressed by many suggested solutions using intermediate pooling processes resembling those in current CNNs. By typically condensing neighborhoods into a single node, graph pooling procedures continuously coarsen the graph in consecutive GNN layers [52, 53]. By removing unnecessary nodes and shortening the distance information must travel, hierarchical coarsening should, in principle, improve long-range learning. No graph pooling method, however, has been discovered to be as broadly applicable as CNN pooling. Modern results are frequently achieved using models that do not coarsen the intermediate graph [54], and some findings imply that neighborhood-local coarsening may be unnecessary or harmful [55]. This study mentions an alternative method for learning long-range dependencies in GNNs and pooling graphs.

This approach is similar to hierarchical pooling in that it substitutes purely learned operations like attention for some of the atomic operations that directly incorporate significant relational inductive biases (e.g., convolutions or spatial pooling in CNNs or neighborhood coarsening in GNNs) [56, 57].

III. METHODOLOGY

Our system, Graph Transformer Networks, is designed to concurrently create new graph structures and learn node representations on the learned graphs. Using several candidate adjacency matrices, GTNs search for novel graph topologies to execute more efficient graph convolutions and learn more potent node representations, in contrast to conventional CNNs on graphs that assume the graph is supplied. Finding meaningful multi-hop connections and meta-paths connected with different types of edges is a necessary step in learning new graph architectures. A brief review of the fundamental ideas of meta-paths and graph convolution in GCNs before proposing this methodology.

A. Preliminaries

Multiple graph architectures with various sorts of nodes and edges are one input to this system. Let T^v and T^e represent the corresponding sets of node types and edge types. It is possible to think of the input graphs as heterogeneous graphs [59]. G = (V, E), where V is a set of observed nodes and E is a set of edges, with a node type mapping function of f_v: V → T^v and an edge type mapping function of f_e: E → T^e. Each node $v_i \in V$ only has one type of node, which is $f_v(v_i) \in T^v$. Similarly, $f_e(e_{ij}) \in T^e$ for $e_{ij} \in E$. It becomes a standard graph when $|T^e| = 1$ and $|T^v| = 1$. This essay examines the scenario where $|T^e| > 1$. Let N represent the total number of nodes, or |V|. The set of adjacency matrices $\{A_k\}_{k=1}^{k}$ where K = $|T^e|$ may be used to depict the heterogeneous graph. $A_k \in R^{NxN}$ is an adjacency matrix where $A_k[i, j]$ is non-zero when there is a k-th type edge connecting j to i. It may be expressed more succinctly as a tensor, $A \in R^{NxNxK}$. Additionally, a feature matrix, $X \in R^{NxD}$, denotes that each node has a D-dimensional input feature.

A route on the heterogeneous graph G, denoted by the symbol p, is connected with heterogeneous edges, such as $v_1 \xrightarrow{t_1} v_2 \xrightarrow{t_2} \dots \xrightarrow{t_1} v_{l+1}$, where $t_l \in T^e$ designates a l-th edge type of meta-path. Between nodes v_1 and v_{l+1}, it defines a composite relation R = $t_1 \circ t_2 \dots \circ t_l$, where $R_1 \circ R_2$ signifies the composition of relation R_1 and R_2. The adjacency matrix A_P of the meta-path P is obtained by multiplying adjacency matrices given the composite relation R or the sequence of edge types (t_1, t_2..., t_l).

$$A_p = A_{t1} \dots . A_{t2} A_{t1}$$

Multi-hop connections are included in the concept of meta-path, and in this approach, novel graph topologies are represented by adjacency matrices. For instance, by multiplying A_{AP} and A_{PC}, the meta-path Author-Paper-Conference (APC), which may be written as $A \xrightarrow{AP} P \xrightarrow{PC} C$ creates an adjacency matrix A_{APC}

1) Graph Convolutional Network (GCN)

In this study, end-to-end node classification representations are learned using a graph convolutional network (GCN) [33]. The forward propagation becomes if $H^{(l)}$ is the feature representation of the lth layer in GCNs

$$H^{l+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l}W^{l}\right),$$

where $W^{(l)} \in \mathbf{R}^{dxd}$ is a trainable weight matrix, $D_{ii} = \sum_i \tilde{A}_{ij}$ is the degree matrix of $\tilde{A}$, and $\tilde{A} = \tilde{A} + I \in R^{N \times N}$ is the adjacency matrix A of graph G with additional self-connections. It is clear that the provided network topology determines the convolution operation over the graph, and that the node-wise linear transform $H^{(l)}W^{(1)}$ is the only method for learning it. This means that the convolution layer, which appears on the graph following a node-wise linear transformation, is composed of a fixed convolution followed by an activation function this methodology benefits from the various convolutions, specifically $\tilde{D}^{-1/2}$ $\tilde{D}^{-1/2}$, acquired from learned multiple adjacency matrices as per graph topologies. It is possible to normalize $\tilde{A}$in (2) for a directed graph (i.e., an asymmetric adjacency matrix) by taking the inverse of the in-degree diagonal matrix D^{-1} as $H^{(l+1)} = \sigma(D^{-1}AH^{(l)}W^{(l)})$.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue I Jan 2023- Available at www.ijraset.com

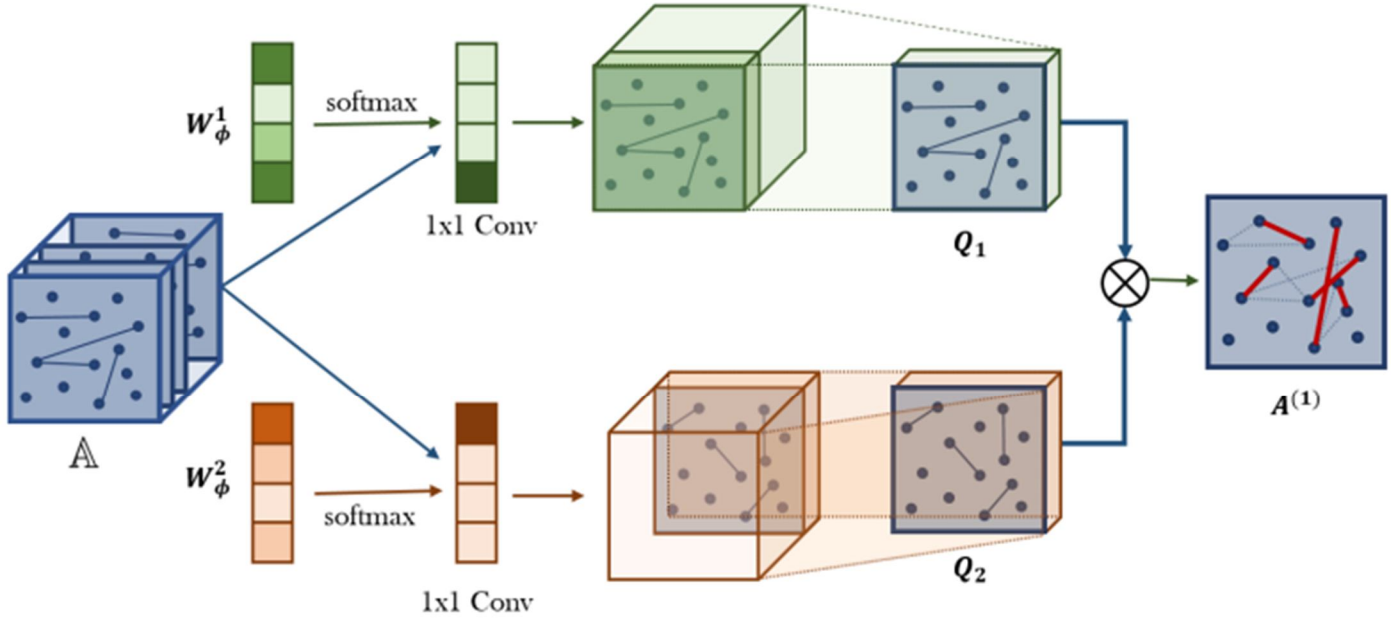


Figure 1 A heterogeneous graph G's adjacency matrices set A is gently picked by the Graph Transformer Layer, which then learns a new meta-path graph represented by A (1) by multiplying the two adjacency matrices it softly chooses, Q1 and Q2, respectively. The selection of the soft adjacency matrix is a weighted sum of the candidate adjacency matrices formed by convolution of 1 x 1 with non-negative weights from softmax ($W_\emptyset^1$).

B. Meta-Path Generation

The meta-path graphs in earlier works [40, 41] must be manually created to apply Graph Neural Networks to them. Instead, Graph Transformer Networks (GTNs) apply graph convolution on the acquired meta-path graphs after learning meta-paths for the provided data and tasks. This provides an opportunity to discover additional practical meta-paths and can result in practically different graph convolutions employing numerous meta-path graphs. There are two parts to the new meta-path graph creation in the Graph Transformer (GT) Layer in Figure 1. The GT layer first softly chooses two candidate adjacency matrices A, and two graph structures, Q_1 and Q_2. Additionally, it picks up a new graph structure by composing two relations (i.e., matrix multiplication of two adjacency matrices, Q_1Q_2).

With the weights from the softmax function, it computes the convex combination of adjacency matrices as $\sum_{t_1 \in T^e} \alpha_{t1}^{(l)} A_{t_1}$ in (4) via 1x1 convolution as shown in Fig.

$$Q = F(\mathbb{A}; W_\emptyset) = \emptyset\left(\mathbb{A}; softmax(W_\emptyset)\right),$$

where $W_\emptyset \in R^{1x1xK}$ is the parameter of Ø and is the convolution layer. Channel attention pooling for low-cost image/action recognition in [60] is comparable to this method. The meta-path adjacency matrix is created by multiplying two adjacency matrices, Q_1 and Q_2, which are carefully selected. The matrix is normalized by its degree matrix as $A^{(l)} = D^{-1}Q_1Q_2$ to maintain numerical stability. The next step is to determine if GTN can learn a meta-path with any edge type and length. One may determine the adjacency matrix of arbitrary length l meta-paths by

$$A_p = \left(\sum_{t_1 \in T^e} \alpha_{t1}^1 A_{t1})\right)\left(\sum_{t_2 \in T^e} \alpha_{t2}^1 A_{t2})\right) \ldots \left(\sum_{t_1 \in T^e} \alpha_{t1}^1 A_{t1})\right),$$

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue I Jan 2023- Available at www.ijraset.com

where α_{t1}^{1} is the weight for edge type t_l at the lth GT layer, T^e signifies a collection of edge types, and A_P denotes the adjacency matrix of meta-paths. A_P may be thought of as the weighted sum of all length-l meta-path adjacency matrices when is not a one-hot vector. Thus, learning any length l meta-path graph topologies is possible using a stack of l GT layers, as demonstrated in the GTN architecture in Fig. 2. This architecture has a problem in that adding GT layers always lengthens the meta-path, making it impossible to use the original edges. Both lengthy and short meta-paths are significant in some applications. Identity matrix I was in cooperated within A, i.e., A_0 = I, to learn short and long meta-paths with original edges. When l GT layers are layered, this method enables GTNs to learn any length of meta-paths up to l+1.

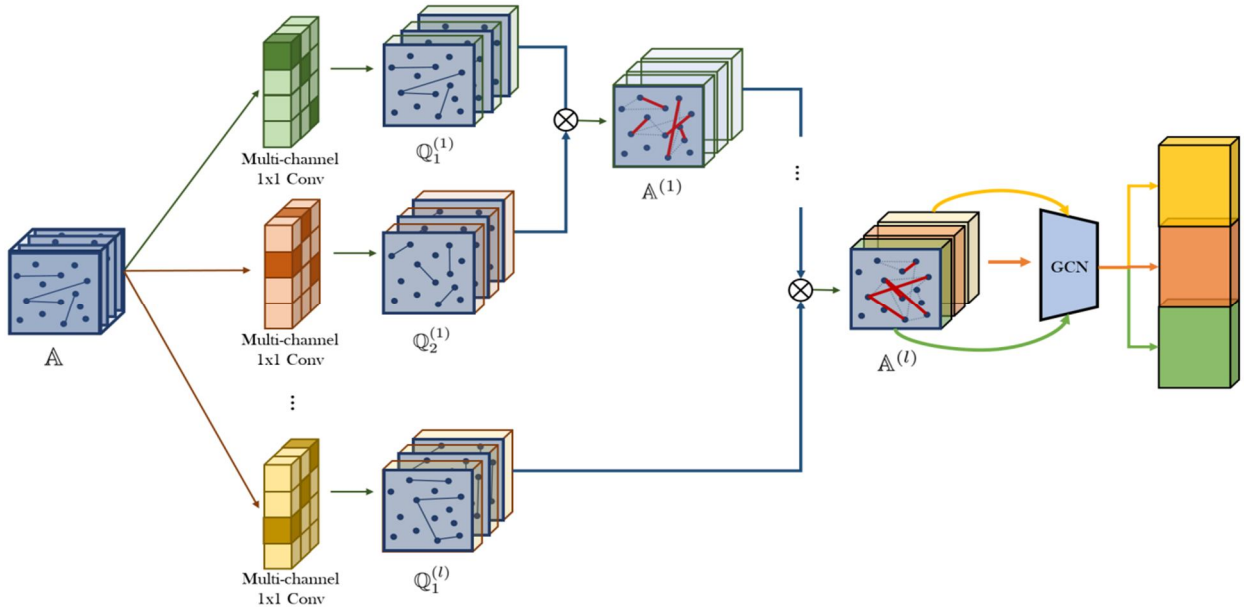


Figure 2 A collection of new meta-path adjacency matrices $A^{(l)}$ are created using Graph Transformer Networks (GTNs) utilizing GT layers, and the new graph structures are then subjected to graph convolution as in GCNs. Concatenating numerous node representations from the same GCNs on various meta-path graphs enhances the performance of node categorization. The intermediate adjacency tensors Q_1^l *and* $Q_2^l \in R^{NxNxC}$ are used to construct meta-paths at the lth layer.

C. Graph Transformer Networks

This section describes the Graph Transformer Networks' architecture. The output channels of the 1x1 convolution in Fig. 1 are set to C to allow for the simultaneous consideration of different sorts of meta-paths. The intermediate adjacency matrices Q_1 and Q_2 then become adjacency tensors Q_1 and Q_2 R^{NxNxC} as seen in Fig. 2 after the GT layer produces a set of meta-paths. Learning various node representations via various graph topologies is advantageous. Multiple node representations are concatenated as a result of applying a GCN to each channel of the meta-path tensor A (l) R^{NxNxC} after the stack of l GT layers.

$$Z = \Big\|_{i=1}^{C} \sigma(\tilde{D}_i^{-1}\tilde{A}_i^{(l)}XW)$$

where W ∈ R^{dxd} is a trainable weight matrix shared across channels, X ∈ R^{Nxd} is a feature matrix, C marks the number of channels, $\tilde{A}_i^l = A_i^l + I$ is the adjacency matrix from the ith channel of $A^{(l)}$, $\tilde{D}_i$ is the degree matrix of $\tilde{A}_i^l$ and || is the concatenation operator. Z is made up of node representations from C distinct meta-path networks, each with various lengths (up to l + 1) Two dense layers, then a softmax layer, are utilized for node classification on top of it. Standard cross-entropy is used as a loss function for the nodes that contain ground truth labels. This architecture may be thought of as an ensemble of GCNs learned from various meta-path graphs.

In this part, a comparison of the advantages of the current approach to several cutting-edge node categorization methods is shown Table 1. Tests and investigations to address the following questions:

Q1. Are the new GTN-generated graph structures useful for teaching node representation?

Q2. Can GTN build meta-paths with varying lengths based on datasets?

Q3. How may the adjacency matrix produced by GTNs be used to understand the significance of each meta-path?

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue I Jan 2023- Available at www.ijraset.com

Table 1 Node classification datasets for heterogeneous networks.

Dataset	Edges	Edge Type	Nodes	Training	Test	Features	Validation
ACM	25922	4	8994	600	2125	1902	300
DBLP	67946	4	18405	800	2857	334	400
IMDB	37288	4	12772	300	2339	1256	300

1) Datasets

Heterogeneous graph datasets were used to formulate relations with a variety of node and edge types to assess the efficacy of meta-paths produced by Graph Transformer Networks. The primary goal is classifying nodes. Use of the IMDB movie dataset was made as well as the DBLP and ACM citation network databases. Table 1 displays the statistics of the heterogeneous graphs utilized in investigations. The DBLP has four types of edges (PA, AP, PC, CP), three types of nodes (papers (P), authors (A), and conferences (C), as well as labels for the authors' study fields. ACM has three different node types (papers (P), authors (A), and topic (S), four different edge kinds (PA, AP, PS, and SP), and paper category labels. The two datasets' nodes are each represented by a bag of words or keywords. On the other side, IMDB has three different node types that correspond to different movie genres: movies (M), actors (A), and directors (D). Plot representations in the form of node characteristics are presented.

2) Implementation Details

For a fair comparison, the embedding dimension was kept fixed for each of the aforementioned approaches at 64. The Adam optimizer was used, and each baseline's best performance was obtained by selecting the appropriate hyperparameters (e.g., learning rate, weight decay, etc.). For models based on random walks, a walk length of 100 per node for 1000 iterations and a window size of 5 with 7 negative samples are used. The validation set is used to optimize the settings for GCN, GAT, and HAN, respectively. Two GT layers were employed for the ACM dataset and three GT layers for the DBLP and IMDB datasets in model GTN. In the GT layer, constant values were used to initialize the parameters for the 1+1 convolution layers.

D. Baselines

GTNs were compared with traditional random walk-based baselines as well as cutting-edge GNN-based approaches to assess the efficacy of representations learned by the Graph Transformer Networks in node categorization.

DeepWalk [44] and metapath2vec [52] have recently demonstrated superior performance among random walk-based techniques among the conventional Network Embedding methods that have been explored. D

eepWalk is a network embedding technique based on random walks that were initially developed for homogenous graphs. Here, DeepWalk was executed on the whole heterogeneous graph while ignoring the node/edge heterogeneity. Metapath2vec, on the other hand, is a heterogeneous graph embedding technique that generates embeddings by skip-gram with negative sampling and meta-path-based random walk.

As GNN-based techniques, GCN [33], GAT [30], and HAN [24] were applied. It uses a localized first-order approximation of the spectral graph convolution created for symmetric graphs. GCN is a graph convolutional network. Degree normalization for asymmetric adjacency matrices was changed, i.e., $\widetilde{D}^{-1}\tilde{A}$ rather than $\widetilde{D}^{-1/2}\tilde{A}\widetilde{D}^{-1/2}$, because datasets are directed networks. The graph neural network known as GAT employs homogeneous graphs to focus its attention mechanism. GCN and GAT were conducted on the entire graph while ignoring the heterogeneity of the nodes and edges. A graph neural network called HAN utilizes explicitly chosen meta-paths. With this method, the main network must be manually divided into smaller graphs by joining vertices with pre-established meta-paths. Here, HAN was put to the test of the chosen sub-graphs whose nodes were connected by meta-paths [24].

IV. RESULT AND DISCUSSION

A representation's ability to be learned from fresh graph configurations. The results of GTN and other node categorization baselines are displayed in Table 2. The response concerning the research questions (Q_1 and Q_2) would be given by analyzing the outcome of the experiment. When compared to all network embedding methods and graph neural network approaches, the proposed GTN gets the top performance on all datasets.

Table 2 Evaluation results on the node classification task (F1 score).

Database	Metapath2vec	GCN	DeepWalk	HAN	GAT	GTN_{-1}	GTN (proposed)
ACM	87.61	91.60	67.42	90.96	92.33	91.13	92.68
DBLP	85.53	87.30	63.18	92.83	93.71	93.91	94.18
IMBD	35.21	56.89	32.08	56.77	58.14	52.33	60.92

Network embedding techniques based on random walks don't perform as well as GNN-based techniques like GCN, GAT, HAN, and GTN. Additionally, the GAT typically outperforms the GCN. This is so that, in contrast to the GCN, which only averages neighbor nodes, the GAT may provide various weights to neighbor nodes. Though the HAN is a modified GAT for a heterogeneous graph, it's interesting to note that the GAT typically outperforms the HAN. This outcome demonstrates that performance issues may arise when utilizing pre-defined meta-paths as the HAN. Contrarily, even though the GTN model only utilizes one GCN layer whereas GCN, GAT, and HAN all require at least two layers, it outperformed all other baselines on all datasets.
It shows that the GTN is capable of learning a new graph structure that has practical meta-paths for mastering more efficient node representation. The GTN can provide edges with different weights as compared to a straightforward meta-path adjacency matrix with constant baselines, such as HAN. To learn variable-length meta-paths, identify the matrix in A. The identity matrix is a part of the candidate adjacency matrices A, as was described in Section 3.2. A different model called $GTN_{-1\ was}$ trained and assessed as an ablation study to confirm the impact of the identity matrix. Although the GTN_{-1} model structure is identical to that of the GTN, it lacks an identity matrix in its candidate adjacency matrix A. The GTN_{-1} routinely performs worse than the GTN in most situations. It is important to notice that IMDB has a bigger difference than DBLP. The length of the meta-paths that GTN_{-1} created may be one reason why they are ineffective in IMDB. Three layers of GTL were layered, and GTN_{-1} consistently produced 4-length meta-paths. However, in IMDB, shorter meta-paths (like MDM) are preferred.

E. Interpretation of Graph Transformer Networks

To explain the interpretability of question Q3, a closer look at the transformation that GTNs is necessary. It is necessary to determine each meta-relevance path using data from GT layers. Suppose there is just one output channel for the sake of simplicity. For a convex combination of input adjacency matrices, create a shorthand notation $\alpha . A = \sum_k^K \alpha_k A_k$ to reduce notational clutter. The output adjacency matrices $A^{(l-1)}$ and input adjacency matrices $\alpha^{(l)}$ of the preceding layer are used by the lth GT layer in Fig. 2 to create an adjacency matrix $A^{(l)}$ for a new meta-path graph.

$$A^l = (D^{l-1})^{-1} A^{l-1} \sum_i^K \alpha_1^l A_i,$$

where A_i is the input adjacency matrix for an edge type i, $D^{(l)}$ is the degree matrix of $A^{(l)}$, and α_i is the weight of A_i. Define $\alpha^{(0)}$ = softmax(W_ϕ^1), $\alpha^{(1)}$ = softmax(W_ϕ^2) since there are two convex combinations at the first layer, as shown in Fig. 1. In GTN, Ql 1 is calculated using the meta-path tensor from the prior tensor, while Q_2^1 is calculated using simply the formula $\alpha^{(1)}$ = softmax(W_ϕ^2) for each layer. The new adjacency matrix from the lth GT layer may then be expressed as follows.

$$A^l = (D^{l-1})^{-1} \dots (D^l)^{-1}((\alpha^0 . \mathbb{A})(\alpha^1 . \mathbb{A})(\alpha^2 . \mathbb{A}) \dots (\alpha^l . \mathbb{A})),$$

$$A^l = (D^{l-1})^{-1} \dots (D^l)^{-1}\left(\sum_{t_0, t_1 \dots t_l \in T^e} \alpha_{t0}^{(0)} \alpha_{t1}^{(1)} \dots \alpha_{tl}^{(l)} A_{t0} A_{t1} \dots A_{tl}\right),$$

where α_{t1}^l is an attention score for edge type t_l at the lth GT layer and T^e signifies a collection of edge types? The weighted total of all meta-paths, from 1-length (original edges) through l-length meta-paths, is thus $A^{(l)}$. $\prod_{i=0}^l \alpha_{ti}^i$ yields the contribution of a meta-path t_l, t_{l-1}..., t_0.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue I Jan 2023- Available at www.ijraset.com

Table 3 Comparison of the top-ranked meta-paths by GTNs and preset meta-paths model discovered significant meta-paths between target nodes that are compatible with pre-defined meta-paths (a type of nodes with labels for node classifications). Additionally, GTNs find fresh relevant meta-paths among all kinds of nodes..

Dataset	Predefined Meta-path	Meta path learned by GTNs	
		Top 3 (between target nodes)	Top 3 (all)
ACM	PAP, PSP	PAP, PSP	APAP, APA, SPAP
DBLP	APCPA, APA	APCPA, APAPA, APA	CPCPA, APCPA, CP
IMBD	MAM, MDM	MDM, MAM, MDMDM	DM, AM, MDM

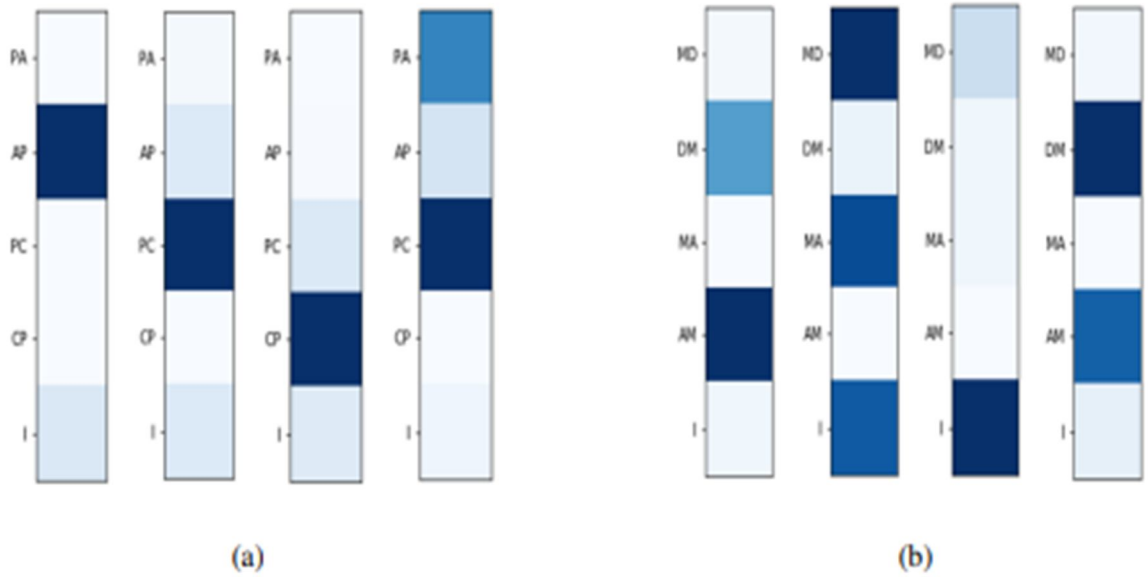


Figure 3 The attention score of the adjacency matrix (edge type) in the DBLP (left) and IMDB (right) datasets were displayed after the softmax function was applied to the 1x1 conv filter Wi I index of the layer) in Figure 1. (A) Each edge in turn denotes the identity matrix, (Paper-Author), (Author-Paper), (Paper-Conference), and (Paper-Conference). (b) The edges in the IMDB dataset show the identification matrix, movie-director, actor-movie, actor-actor, and movie-actor relationships.

Now that GTNs have learned new graph topologies, they can be read. For a meta-path (t_0, t_1... t_l), the weight $\prod_{i=0}^{l} \alpha_{ti}^{i}$ is an attention score that indicates how significant the meta-path is to the prediction job. Table 3 lists the predetermined meta-paths that are often used in literature as well as the meta-paths that GTNs learned that had high attention ratings. The prepared meta-paths by domain knowledge are regularly rated first by GTNs between target nodes that contain class labels to forecast, as shown in Table 3. This demonstrates that GTNs can pick up on the significance of meta-paths for jobs. What's more intriguing is that GTNs found significant meta-paths that weren't already part of the established meta-path set.

For instance, CPCPA, which is not part of the present meta-path collection, is ranked as the most important meta-path in the DBLP dataset by GTN. It makes it reasonable that the author's published works would be related to the author's study field (label to forecast).

The interpretability of GTNs offers helpful information on the categorization of nodes based on attention ratings on meta-paths. The attention ratings of the adjacency matrices (edge type) from each Graph Transformer Layer have displayed in Fig. 3. Identity matrices had greater attention scores in IMDB as compared to the DBLP results. A GTN, which is more useful than IMDB, can learn shorter meta-paths than the number of GT layers, as was stated in Section 3.3. The GTN tries to cling to the shorter meta-paths even in the deeper layer by giving the identity matrix greater attention ratings. This finding shows that the GTN can learn the best meta-path length based on the dataset in an adaptable manner.

V. CONCLUSION

For learning node representation on a heterogeneous graph, this research introduced Graph Transformer Networks. This method learns node representation by convolution on the learned meta-path graphs while transforming a heterogeneous network into numerous new graphs defined by meta-paths with variable edge types and lengths up to one fewer than the number of Graph Transformer layers. With no specified meta-paths from domain knowledge, the learned graph topologies produce a state-of-the-art performance on all three benchmarks for node classification on heterogeneous networks. It's observed that this framework creates new opportunities for GNNs to optimize graph structures on their own to operate convolution based on data and jobs without any human efforts since these Graph Transformer layers can be integrated with current GNNs. Future research should examine the effectiveness of GT layers paired with various kinds of GNNs rather than GCNs. Applying GTNs to the other tasks might be interesting future paths as well, since various heterogeneous graph datasets have recently been examined for other network research tasks, such as link prediction [61,62] and graph categorization [63, 64].

REFERENCES

[1] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. ArXiv preprint, abs/1806.01261, 2018. URL https://arxiv.org/abs/1806.01261.

[2] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In Advances in neural information processing systems, 5998–6008.

[3] Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 .

[4] Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, 1263–1272. JMLR. org.

[5] Joshi, C. 2020. Transformers are Graph Neural Networks. The Gradient .

[6] Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 .

[7] Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

[8] Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165 .

[9] Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In European Semantic Web Conference, 593–607. Springer.

[10] Chami, I.; Wolf, A.; Juan, D.-C.; Sala, F.; Ravi, S.; and Re,´ C. 2020. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. arXiv preprint arXiv:2005.00545 .

[11] Monti, F.; Frasca, F.; Eynard, D.; Mannion, D.; and Bronstein, M. M. 2019. Fake news detection on social media using geometric deep learning. arXiv preprint arXiv:1902.06673 .

[12] Cranmer, M. D.; Xu, R.; Battaglia, P.; and Ho, S. 2019. Learning Symbolic Physics with Graph Networks. arXiv preprint arXiv:1909.05862 .

[13] Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; and Battaglia, P. W. 2020. Learning to simulate complex physics with graph networks. arXiv preprint arXiv:2002.09405 .

[14] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with Region Rroposal Networks. In Advances in Neural Information Processing Systems (pp. 91–99).

[15] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 1097–1105).

[16] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 580–587).

[17] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (p. 815-823).

[18] Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., & Chua, T.-S. (2017). SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 5659–5667).

[19] Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performanceoptimized hierarchical models predict neural responses in higher visual cortex. Proceedings of the National Academy of Sciences, 111(23), 8619–8624.

[20] Geirhos, R., Meding, K., & Wichmann, F. A. (2020). Beyond accuracy: Quantifying trial-by-trial behaviour of CNNs and humans by measuring error consistency. In Advances in Neural Information Processing Systems

[21] Hermann, K., Chen, T., & Kornblith, S. (2020). The origins and prevalence of texture bias in convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 19000–19015).

[22] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In Advances in neural information processing systems, pages 2224–2232, 2015.

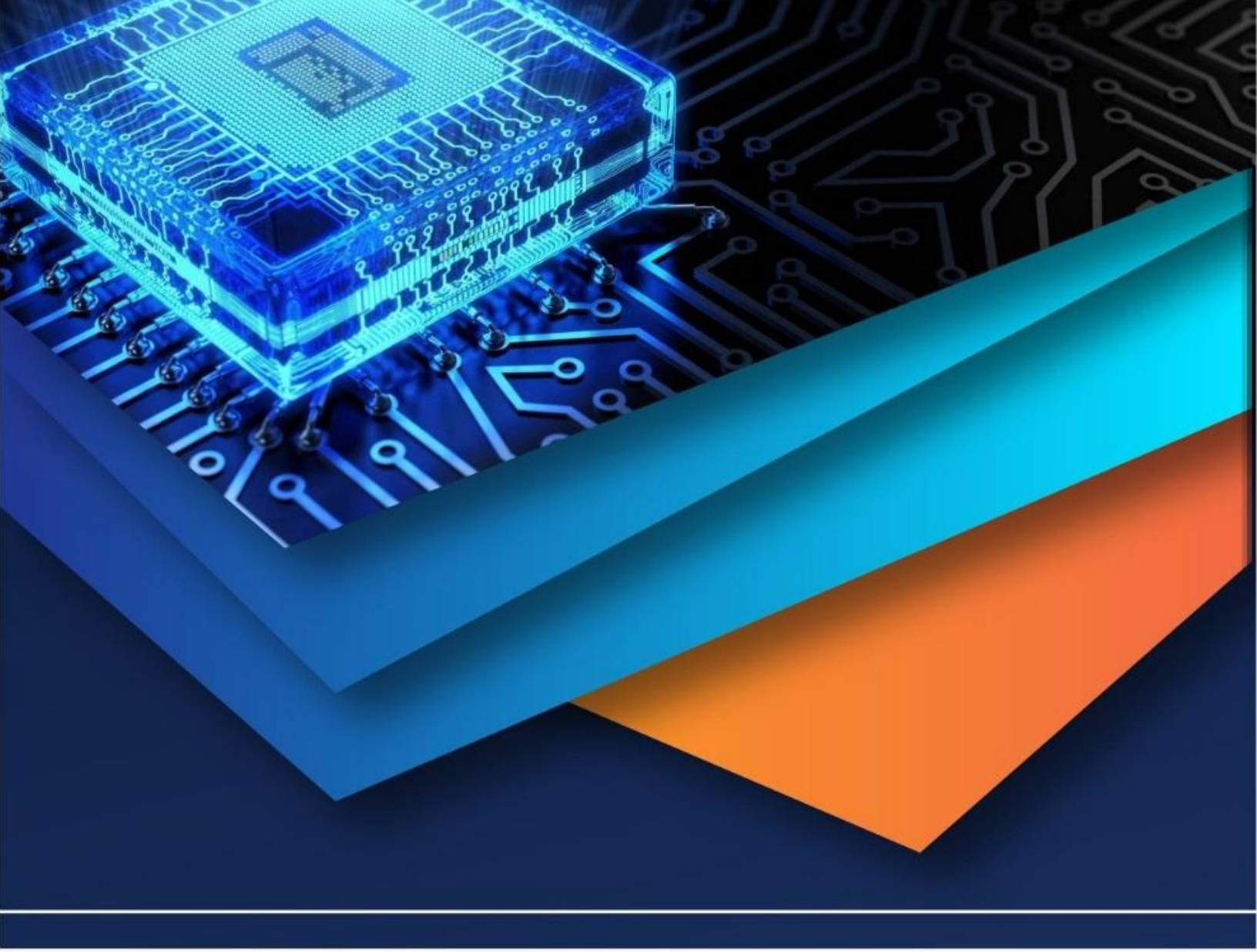
[23] J. Lee, I. Lee, and J. Kang. Self-attention graph pooling. CoRR, 2019.

[24] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. CoRR, abs/1806.08804, 2018.

[25] T. N. Kipf and M. Welling. Variational graph auto-encoders. NIPS Workshop on Bayesian Deep Learning, 2016.

[26] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In European Semantic Web Conference, pages 593–607, 2018. [31] C. Shi, Y. Li, J. Zhang, Y
[27] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In Advances in Neural Information Processing Systems, pages 5165–5175, 2018.
[28] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In Social network data analytics, pages 115–148. 2011.
[29] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. CoRR, abs/1706.02216, 2017.
[30] P. Velickovi ˇ c, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention ´ networks. In International Conference on Learning Representations, 2018. URL https: //openreview.net/forum?id=rJXMpikCZ.
[31] J. Chen, T. Ma, and C. Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In International Conference on Learning Representations, 2018.
[32] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1225–1234, 2016.
[33] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.
[34] S. I. Ktena, S. Parisot, E. Ferrante, M. Rajchl, M. C. H. Lee, B. Glocker, and D. Rueckert. Distance metric learning using graph convolutional networks: Application to functional brain networks. CoRR, 2017.
[35] R. v. d. Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263, 2017.
[36] F. Monti, M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multigraph neural networks. In Advances in Neural Information Processing Systems, pages 3697– 3707, 2017.
[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 974–983, 2018.
[38] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in neural information processing systems, pages 3844–3852, 2016.
[39] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. CoRR, abs/1506.05163, 2015.
[40] X. Wang, H. Ji, C. Shi, B. Wang, P. Cui, P. Yu, and Y. Ye. Heterogeneous graph attention network. CoRR, abs/1903.07293, 2019.
[41] Y. Zhang, Y. Xiong, X. Kong, S. Li, J. Mi, and Y. Zhu. Deep collective classification in heterogeneous information networks. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, pages 399–408, 2018.
[42] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013.
[43] J. Chen, J. Zhu, and L. Song. Stochastic training of graph convolutional networks with variance reduction. arXiv preprint arXiv:1710.10568, 2017.
[44] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 701–710, 2014.
[45] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. Journal of Machine Learning Research, 11(Apr):1201–1242, 2010.
[46] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. Journal of the American society for information science and technology, 58(7):1019–1031, 2007.
[47] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In WWW, 2015.
[48] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[49] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 34(4):18–42, 2017.
[50] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. CoRR, abs/1611.08402, 2016.
[51] Z. Liu, C. Chen, L. Li, J. Zhou, X. Li, L. Song, and Y. Qi. Geniepath: Graph neural networks with adaptive receptive paths. arXiv preprint arXiv:1802.00910, 2018.
[52] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In KDD '17, pages 135–144, 2017.
[53] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 3837–3845, 2016.
[54] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 4805–4815, 2018.
[55] V. Thost and J. Chen. Directed acyclic graph neural networks. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021.
[56] D. P. P. Mesquita, A. H. S. Jr., and S. Kaski. Rethinking pooling in graph neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.
[57] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-End Object Detection with Transformers. ArXiv preprint, abs/2005.12872, 2020.
[58] J. Cordonnier, A. Loukas, and M. Jaggi. On the relationship between self-attention and convolutional layers. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020.
[59] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip. A survey of heterogeneous information network analysis. IEEE Transactions on Knowledge and Data Engineering, 29(1):17–37, 2016.

[60] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. A2-nets: Double attention networks. In ˆ Advances in Neural Information Processing Systems, pages 352–361, 2018.
[61] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua. Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pages 950–958, 2019.
[62] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pages 793–803, 2019.
[63] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li. Heterogeneous graph attention networks for semi-supervised short text classification. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019.
[64] R. Kim, C. H. So, M. Jeong, S. Lee, J. Kim, and J. Kang. Hats: A hierarchical graph attention network for stock movement prediction, 2019.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY