GraphInception: Convolutional Neural Networks for Collective Classification in Heterogeneous Information Networks

Yun Xiong, Yizhou Zhang, Xiangnan Kong, Huidi Chen, and Yangyong Zhu

Abstract—Collective classification has attracted considerable attention in the last decade, where the labels within a group of instances are correlated and should be inferred collectively, instead of independently. Conventional approaches on collective classification mainly focus on exploiting simple relational features (such as *count* and *exists* aggregators on neighboring nodes). However, many real-world applications involve complex dependencies among the instances, which are obscure/hidden in the networks. To capture these dependencies in collective classification, we need to go beyond simple relational features and extract deep dependencies between the instances. In this paper, we study the problem of deep collective classification in *Heterogeneous Information Networks* (HINs), which involve different types of autocorrelations, from simple to complex relations, among the instances. Different from conventional autocorrelations, which are given explicitly by the links in the network, complex autocorrelations are obscure/hidden in HINs, and should be inferred from existing links in a hierarchical order. This problem is highly challenging due to the multiple types of dependencies among the nodes and the complexity of the relational features. In this study, we proposed a deep convolutional collective classification method, called *GraphInception*, to learn the deep relational features in HINs. And we presented two versions of the models with different inference styles. The proposed methods can automatically generate a hierarchy of relational features with different complexities. Extensive experiments on four real-world networks demonstrate that our approach can improve the collective classification performance by considering deep relational features in HINs.

Index Terms—Collective classification, graph convolution, heterogeneous information networks, graph mining, deep learning

1 Introduction

Collective classification [1] aims at predicting the class labels of the unlabeled nodes according to the graph and the labels of neighboring nodes [2], [3]. In many relational data, the labels of different instances can be related. For example, in bibliographic networks, the papers written by the same author are more likely to share similar topics than those written by different authors. An effective model for relational data should be able to capture the dependencies among different instances and perform classification collectively. Motivated by this challenge, collective classification has been extensively studied in recent years [1], [4], [5], [6].

Previous works on collective classification are concentrated on shallow models, which depend heavily on the design of relational features by the experts. On one hand, conventional relational features are usually defined as a simple aggregation of a node's direct neighbors, such as the average or the count of their labels. On the other hand, recent works on deep learning models [7] offer automatic end-to-end feature learning in a variety of domains, such as vision [8], speech and NLP [9]. However, deep learning models mainly focus on content features, e.g., visual features in image data. They have not yet been used to extract deep relational features in collective classification, to capture the complex autocorrelation among instances in relational learning.

In this paper, we study the problem of *deep convolutional collective classification* in HINs, which involves a hierarchy of different types of autocorrelations among the instances. For example, predicting the research area (label) of the authors in *bibliographic networks*, as showed in Fig. 1. Different authors are not only explictly inter-connected through co-author relations, but also implicitly connected through latent relations, such as adviser-advisee, share-advisor, colleague. The dotted lines in Fig. 1 represent implicit relationships between authors that exist in real world, but can only be inferred in the DBLP network. Although there exist shallow methods on collective classification in HINs [10], [11], it is still an open and challenging problem due to the following reasons:

- *Deep Relational Features:* One major challenge of the problem is that HINs can involve a hierarchy of different types of autocorrelations, from simple to complex ones, among the instance. The complex relationships are not given directly by the links in the

- *Y. Xiong, Y. Zhang, H. Chen, and Y. Zhu are with the Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, Shanghai Institute for Advance Communication and Data Science, Fudan University, Shanghai 200433, China. E-mail: {yunx, yizhouzhang14, yyzhu}@fudan.edu.cn, jjj96178@163.com.*
- *X. Kong is with the Department of Worcester Polytechnic Institute, Worcester, MA 01609-2280 USA. E-mail: xkong@wpi.edu.*

Manuscript received 13 June 2018; revised 18 July 2019; accepted 6 Oct. 2019. Date of publication 15 Oct. 2019; date of current version 1 Apr. 2021.
(Corresponding author: Yun Xiong.)
Recommended for acceptance by K. Selcuk Candan.
Digital Object Identifier no. 10.1109/TKDE.2019.2947458

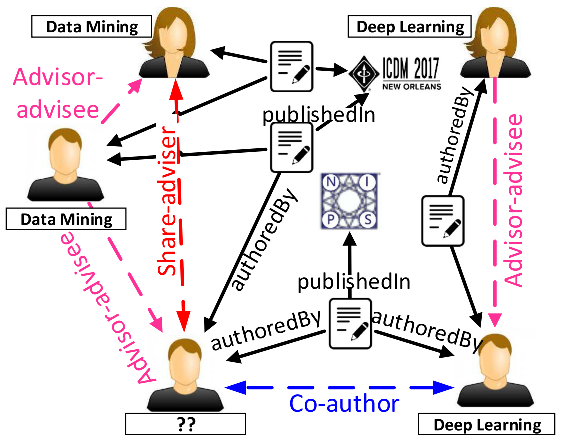


Fig. 1. An example of *simple* links and *hidden* links in DBLP network. Here, the solid lines represent the *simple* relations exist in DBLP network, and the dotted lines represent the *hidden* relations that exist in real world but should be inferred from DBLP network, i.e., co-author (blue line), adviser-advisee (pink line), and share-adviser (red line).

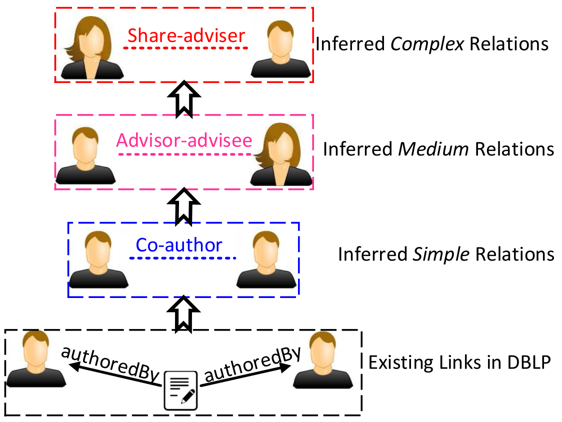

Share-adviser
Inferred *Complex* Relations
Advisor-advisee
Inferred *Medium* Relations
Co-author
Inferred *Simple* Relations
authoredBy
authoredBy
Existing Links in DBLP

Fig. 2. An example of a hierarchical relations between authors in the DBLP network. Here, the complex/obscure relations (above) can be inferred from the existing/simple relations (bottom).

network, but can be inferred by a hierarchy of relational features. For instance, in the DBLP network in Fig. 1, there are co-author relations (simple relationships) between authors; advisor-advisee relations (median relationships), which can be inferred from co-author relations; and share-adviser relations (complex relationships), which are deeply hidden in the network, but can be inferred from advisor-advisee relations. The complex relations, e.g., share-advisor, cannot be directly modeled by shallow relational features like co-author relations, but can potentially be inferred from a hierarchy of deep relational features, from simple ones (co-authors), medium ones(advisor-advisee), to complex ones (share-advisor), as shown in Fig. 2. For example, to infer the advisor-advisee relationships, we could find the two authors sharing similar neighboring nodes (these neighbors are most likely to be their adviser and other students in the research group). Because of the complex and obscure relationships between the instances in heterogeneous networks, we need a deep relational learning model to extract a hierarchy of deep dependencies among the instances.

- *Mixed Complexity in Relational Features:* The second major challenge of the problem is the diversity of the complexity levels in the relational features. As shown in Fig. 1, there are usually a mixture of both simple and complex dependencies among the instances, which can all be related to the collective classification task. In these networks, a simple relational model can only capture simple relations, but will be underfitting on complex relations. On the other hand, a conventional deep learning model with deep layers may only capture complex relations, but will be overfitting on simple relations. Therefore, an ideal model should be able to automatically balance its model complexity with respect to the mixture of complexities.
- *Heterogeneous Dependencies:* The third major challenge with applying deep learning models on collective classification lies in the diversity of the node types and link types in HINs. The properities of different types of nodes (links) are very different, which makes it difficult to apply deep learning models directly. For example, graph convolution models [12], assume each node in the network sharing same convolution kernels, which is untenable in HINs. Although there exist some researches for collective classification in HINs [10], [11], however, most of them are shallow models which ignore the deep relational features in the network.
- *Efficiency in Inference stage:* The last challenge of the problem is the efficiency in the inference stage. In many real world applications, we usually care more about the efficiency of the inference stage than the training stage. However, many collective classification models require iterative inference process, and they need to be iterated many times, which can be slow in the inference stage.

In order to address the above challenges, we present a deep graph convolutional model, called GraphInception, for collective classification in HINs. Fig. 3 compares the difference between GraphInception and other conventional methods. Considering the diversity of the complexity levels in the relational features (some are very simple and some are complex), in order to study the relational features more efficiently, we propose the *graph inception module* to balance relational features with different complexities. This model is inspired by the inception module [14], a highly efficient deep convolutional model for CNN with fewer parameters and deeper layers. To our best knowledge, the graph inception module has not been studied in the literature.

2 Preliminaries

Notation convention: We use capital letters for matrices, bold lowercase letters for vectors and handwritten letters for sets. The operator $\otimes$ is used to denote convolution operation. We summarized all notations in Table 1.

2.1 Heterogeneous Information Network

In many real-world applications, the networks include multiple types of nodes and links, which are called *heterogeneous information networks* [15].

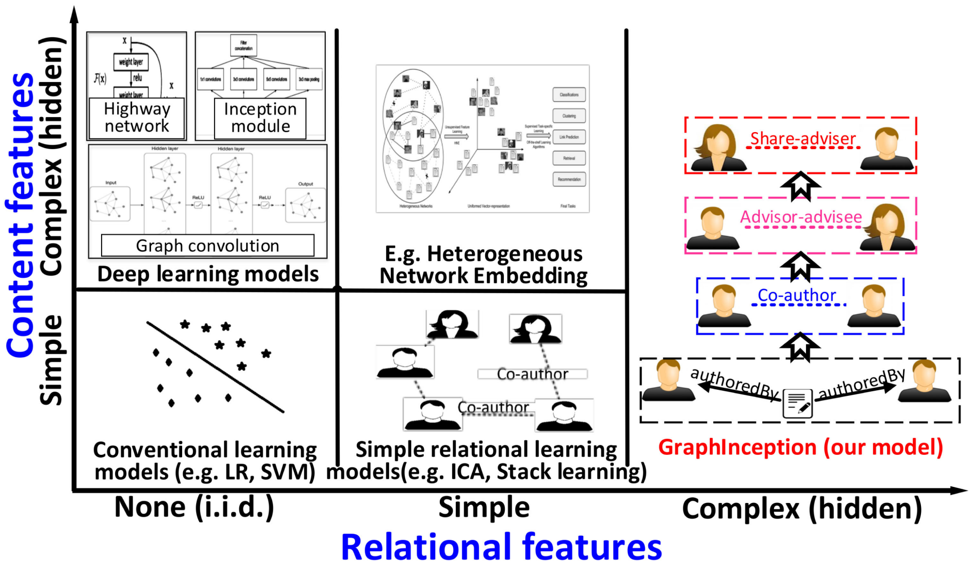


Fig. 3. The difference between the proposed method and conventional methods. We divided the algorithms according to the difficulty of the content features and the relational features. In particular, the deep learning models include highway network [13], inception module [14], and graph convolution model [12].

Definition 1. *Heterogeneous information network is a special kind of information network, which can be represented as a directed graph $G = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ denotes the set of nodes, including m types of nodes: $\mathcal{V}_1 = \{v_{11}, \ldots, v_{1n_1}\}, \ldots, \mathcal{V}_m = \{v_{m1}, \ldots, v_{mn_m}\}$, where v_{ji} represents the ith instance of type j. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the links between the nodes in $\mathcal{V}$, which involves multiple types of links.*

For example, as shown in Fig. 1, the DBLP network includes three types of nodes, e.g., authors, papers, conferences, which are connected through two types of links, e.g., authoredBy, PublishedIn.

2.2 Collective Classification in HINs

In this paper, we focus on studying the collective classification problem on one type of nodes, instead of on all of them in HINs. The reason is the label space of different types of nodes are quite different, so it's unreasonable to assume all types of nodes share the same set of label concepts. For instance, in movie networks, e.g., IMDB [16], the label concepts for movie genres classification task are only defined on movie nodes, instead of director nodes or actor nodes. In a specific inference task, we usually only care about the inference results on one type of nodes.

Without loss of generality, we suppose the first node type $\mathcal{V}_1$ in the HIN G as the type of target nodes we need to inference, and suppose we have n nodes in $\mathcal{V}_1$. On each node $v_{1i} \in \mathcal{V}_1$, we have a features vector $\mathbf{x}_i \in \mathbb{R}^d$ in the d-dimensional space; and we also have a label variable $Y_i \in \{1, \ldots, C\}$ indicating the class label assigned to node v_{1i}. $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\mathcal{Y} = \{Y_1, \ldots, Y_n\}$ represent the set of features and the set of labels for all instances in $\mathcal{V}_1$.

The instances in $\mathcal{V}_1$ are then divided into a training set $\mathcal{L}$ and a test set $\mathcal{U}$, where $\mathcal{L} \bigcup \mathcal{U} = \mathcal{V}_1$ and $\mathcal{L} \bigcap \mathcal{U} = \emptyset$. We use $\mathcal{Y}_\mathcal{L} = \{Y_i | v_{1i} \in \mathcal{L}\}$ represents the labels set of the nodes in the training set, and use $\mathcal{Y}_\mathcal{U} = \{Y_i | v_{1i} \in \mathcal{U}\}$ represents the labels set of the nodes in the test set. Collective inference methods assume that the instances linked in the network are related [1]. Let $\mathcal{N}_i^{(t)}$ ($\mathcal{N}_i^{(t)} \subseteq \mathcal{V}_1$) represents the set of t-order neighbors of node v_{1i}, $\mathcal{Y}_{\mathcal{N}_i^{(t)}} = \{Y_i | v_{1i} \in \mathcal{N}_i^{(t)}\}$. Then the task of *collective classification in HINs* is to estimate:

$$\Pr(\mathcal{Y}_\mathcal{U} | \mathcal{X}, \mathcal{Y}_\mathcal{L}) \propto \prod_{v_{1i} \in \mathcal{U}} Pr(Y_i | \mathbf{x}_i, \mathcal{Y}_{\mathcal{N}_i^{(1)}}, \ldots, \mathbf{x}_{\mathcal{N}_i^{(t-1)}}, \mathcal{Y}_{\mathcal{N}_i^{(t)}}). \tag{1}$$

It is challenging to learn and infer about $Pr(Y_i | \mathbf{x}_i, \mathcal{Y}_{\mathcal{N}_i^{(1)}}, \ldots, \mathbf{x}_{\mathcal{N}_i^{(t-1)}}, \mathcal{Y}_{\mathcal{N}_i^{(t)}})$, especially to learn the deep relational features of the nodes with complex correlations in HIN. In next section, we propose a framework to solve the problem.

TABLE 1
Important Notations

Symbol	Definition
$G = (\mathcal{V}, \mathcal{E})$	A heterogeneous information network.
$\mathcal{V} = \{\mathcal{V}_1, \ldots, \mathcal{V}_m\}$	The set of nodes, involving m types of nodes. The target node type is $\mathcal{V}_1$.
$\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$	The set of local features for all instances in $\mathcal{V}_1$.
$\mathcal{Y} = \{Y_1, \ldots, Y_n\}$	The set of label values for all instances in $\mathcal{V}_1$, $Y_i \in \{1, \ldots, C\}$.
$\mathcal{L}$ and $\mathcal{U}$	The training set and test set, where $\mathcal{L} \bigcup \mathcal{U} = \mathcal{V}_1$, $\mathcal{L} \bigcap \mathcal{U} = \emptyset$.
$\mathcal{S} = \{\mathcal{P}_1, \ldots, \mathcal{P}_{\lvert\mathcal{S}\rvert}\}$	The set of meta paths type. Each $\mathcal{P}_l$ denotes a composite relationship between instances in $\mathcal{V}_1$.
$\mathbf{X}$	The convolutional signal of nodes, where $\mathbf{X} \in \mathbb{R}^{n \times C}$.
K	The convolutional kernel size.
F	Number of convolutional filters, i.e., the hidden dimension of the filter.
$\mathbf{H}^t$ and T	$\mathbf{H}^t$ denotes the convolution result at tth layer in neural networks. T is the number of convolutional layers.

3 Graph Convolution-Based Deep Relational Feature Learning

For learning the deep relational features in HINs, we present the graph convolution-based relational feature learning

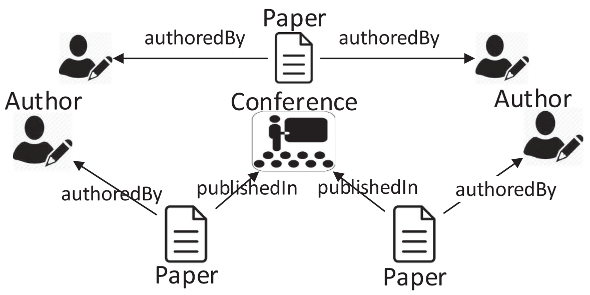


Fig. 4. Examples of different types of meta-path between two authors in DBLP network.

model in this section. The model includes two phases: i) *multi-channel network translation*, which translates the HIN to a multi-channel network so that we can do convolution in the HIN; ii) *graph convolution based relational feature learning*, which learns the deep relational features from multi-channel networks based on graph convolution. The architecture of our method is shown in Fig. 5.

3.1 Multi-Channel Network Translation

There are abundant types of nodes in a HIN, while the diversities between different node types are vary widely, which greatly increased the difficulty of convolution operation on the network. Usually, we only care about one type of nodes, instead of all of them in HINs. To simplify the learning curve, we propose the *multi-channel network*, each channel of which is a homogeneous network consisting of the target nodes type, and the links (relationships) are extracted from the HIN with different semantic meaning. In this subsection, we first introduce a concept named *meta path* [15], which is often used to extract relationships among the instances in HINs. Then we propose how to translate the HIN to a multi-channel network based on meta paths.

The instances in HIN are inter-connected through multiple types of links. Each type of links from node type $\mathcal{V}_i$ to node type $\mathcal{V}_j$ corresponds to a binary relation R, where $R(v_{ip}, v_{jq})$ holds if the node v_{ip} and v_{jq} are linked in R. For example, in Fig. 1, the link type "authoredBy" is a relation between *paper* nodes and *author* nodes, where $R(v_{ip}, v_{jq})$ holds if the *paper* node v_{ip} has a link of type "authoredBy" to the *author* node v_{jq} in the network. We can write the link type as "paper $\xrightarrow{authoredBy}$ author". The *meta path* is defined as a sequence of relations in the network schema. For instance, a meta path "author $\xrightarrow{authoredBy^{-1}}$ paper $\xrightarrow{authoredBy}$ author" denotes a composite relationship between *author* nodes, where the semantic meaning of this meta-path is that the two authors are *Co-author*. Here the link type "authoredBy^{-1}" represents the inverted relation of "authoredBy". We give two types of meta path between *author* nodes in Fig. 4.

Each meta path defines a unique relationship between nodes, and can be used as a link type to a specific channel in the multi-channel network. For learning the dependencies among instances more effectively, we translate the HIN to the multi-channel network, where each channel of the network is connected via a certain type of meta path. Formally, given a set of meta paths $\mathcal{S} = \{\mathcal{P}_1, \ldots, \mathcal{P}_{|\mathcal{S}|}\}$, the translated multi-channel network G' is defined as:

$$G' = \{G'_\ell | G'_\ell = (\mathcal{V}_1, \mathcal{E}_{1\ell}), \ell = 1, \ldots, |\mathcal{S}|.\}, \tag{2}$$

where $\mathcal{E}_{1\ell} \subseteq \mathcal{V}_1 \times \mathcal{V}_1$ denotes the links between the instances in $\mathcal{V}_1$, which are connected through the meta path $\mathcal{P}_\ell$.

There are many ways to construct meta paths: At the beginning, meta paths were constructed manually by the experts; Afterwards, several efficient methods were presented to construct meta paths including using the breadth-first search within a limited path length [10] and greedy tree-based model [17]. In this paper, We choose the breadth-first search to construct meta paths.

3.2 Graph Convolution-based Relational Feature Learning

In this subsection, we first propose the idea of learning the relational features from homogeneous networks based on graph convolution, then extend the idea into HINs.

In this study, we focus on relational feature learning, while conventional graph convolution models mainly focus on content feature learning [12], [18], [19]. We summarized the significant differences between them as shown in Table 2 and the last paragraph in this subsection. Considering the graph convolution on a homogeneous network G_{homo}, the convolution theorem defines convolutions as linear operators that diagonalize in the Fourier basis. We use the eigenvectors of the graph transition probability matrix $\mathbf{P}$ as Fourier basis. Then the convolution on G_{homo} is defined as the multiplication of a

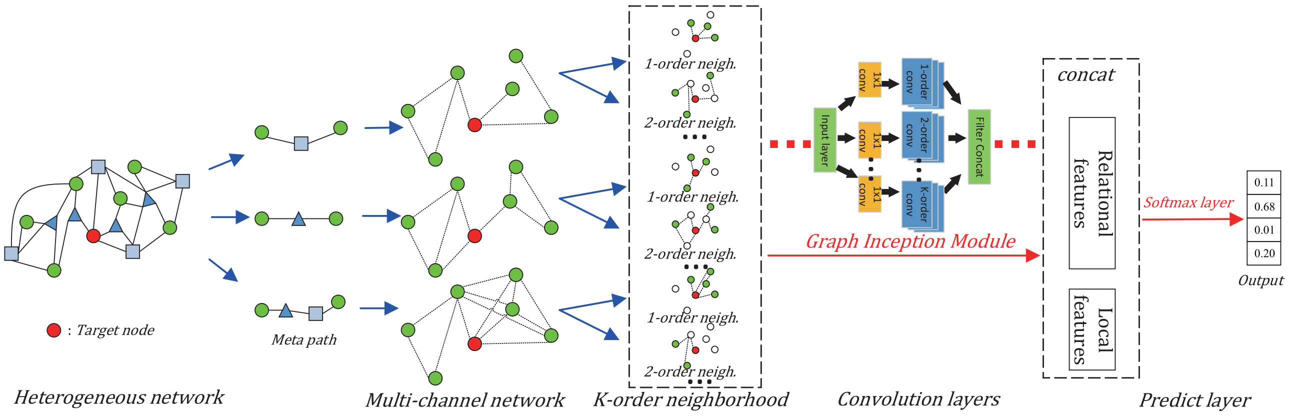


Fig. 5. Architecture of the proposed GraphInception in HINs. The details of the graph inception module is shown in Fig. 6.

TABLE 2
The Differences between Our Model and Graph Convolution Models

	Graph convolution models	Our model
Target problem	Graph classification	Collective classification
Convolution matrix	Laplacian matrix **L**	Transition probability matrix **P**
Convolution Input	Neighbor nodes with itself	Neighbor nodes only
Convolution Output	Content features	Relational features
Networks	Homogeneous networks	Heterogeneous networks

signal $\mathbf{X} \in \mathbb{R}^{n\times C}$ (a C-dimensional feature vectors for each node) with a filter g_θ on $\mathbf{P}$ in the Fourier domain, i.e.,

$$g_\theta \otimes_{G_{\text{homo}}} \mathbf{X} = g_\theta(\mathbf{P})\mathbf{X} = g_\theta(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top)\mathbf{X} = \mathbf{U}g_\theta(\boldsymbol{\Lambda})\mathbf{U}^\top\mathbf{X}, \tag{3}$$

where $\mathbf{U}$ is the eigenvector matrix of $\mathbf{P}$, $\boldsymbol{\Lambda}$ is the diagnal matrix of eigenvalues of $\mathbf{P}$, $g_\theta(\boldsymbol{\Lambda})$ is the vector of Fourier coefficients, and $\mathbf{U}^\top\mathbf{X}$ is the graph Fourier transform of $\mathbf{X}$. Note that $\mathbf{U}$ and $\boldsymbol{\Lambda}$ are complex matrices because $\mathbf{P}$ is unsymmetrical matrix. However, recent work [20] has successfully applied the complex valued in traditional CNN. And we will see later, our model has nothing to do with whether $\mathbf{U}$ is complex matrix.

For selecting the local neighbors of a given node, we define g_θ as a polynomial-parametric filter [12]:

$$g_\theta(\boldsymbol{\Lambda}) = \sum_{k=1}^{K} \theta_k \boldsymbol{\Lambda}^k, \tag{4}$$

where the parameter $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients. Then we have:

$$g_\theta \otimes_{G_{\text{homo}}} \mathbf{X} = \mathbf{U}\sum_{k=1}^{K}\theta_k\boldsymbol{\Lambda}^k\mathbf{U}^\top\mathbf{X} = \sum_{k=1}^{K}\theta_k\mathbf{P}^k\mathbf{X}. \tag{5}$$

Note that this expression is now K-order neighborhood since it is a Kth order polynomial of the transition probability matrix, i.e., it depends only on nodes that are at maximum K steps away from the target node.

Now, we extend the Eq. (5) to HINs. Previously, we introduced how to translate a HIN G to a multi-channel network G', where each channel of the network represents a particular relationship between the nodes in $\mathcal{V}_1$. In this subsection, we learned the relational features on HIN via convolution on each channel of G', i.e.,

$$\begin{aligned} g_\theta \otimes_G \mathbf{X} &= \left(g_{\theta_1} \otimes_{G'_1} x, \ldots, g_{\theta_{|\mathcal{S}|}} \otimes_{G'_{|\mathcal{S}|}} \mathbf{X}\right) \\ &= \left(g_{\theta_1}(\mathbf{P}_1)\mathbf{X}, \ldots, g_{\theta_{|\mathcal{S}|}}(\mathbf{P}_{|\mathcal{S}|})\mathbf{X}\right), \end{aligned} \tag{6}$$

where $\mathbf{P}_\ell$ $(\ell = 1, \ldots, |\mathcal{S}|)$ represents the transition probability matrix of G'_ℓ. Note that we use different convolutional filters on different channels, and finally concat the convolution results. The reason is that the nodes have different neighbor nodes in each channel, thus are not suitable for convolution on all channels with one filter. Moreover, we generalize Eq. (6) to F filters for feature maps, i.e., the hidden dimension of the convolutional filter is F. Then we have:

$$\begin{aligned} \mathbf{H} = g_\Theta \otimes_G \mathbf{X} &= \left(g_{\Theta_1}(\mathbf{P}_1)\mathbf{X}, \ldots, g_{\Theta_{|\mathcal{S}|}}(\mathbf{P}_{|\mathcal{S}|})\mathbf{X}\right) \\ &= r\left(\sum_{k=1}^{K}\mathbf{P}_1^k\mathbf{X}\Theta_{1k}, \ldots, \sum_{k=1}^{K}\mathbf{P}_{|\mathcal{S}|}^k\mathbf{X}\Theta_{|\mathcal{S}|k}\right), \end{aligned} \tag{7}$$

$\Theta_{\ell k} \in \mathbb{R}^{C\times F}$ $(\ell = 1, \ldots, |\mathcal{S}|)$ is now a matrix of filter parameters. A function r of a vector $\mathbf{x}$ is defined as $r(\mathbf{x}) = (r(x_1), \ldots, r(x_n))$, here $r(x_i) = \max(0, x_i)$ is the Relu function. The ith row vector of $\mathbf{H}$ represents the learned relational features of the node v_{1i}.

In summary, the differences between conventional graph convolution models and our model include the following aspects: 1) Conventional models concentrate on graph classification problem, while our model focus on collective classification problem; 2) Most graph convolution models are content feature learning models, therefore, they use the eigenvectors of the laplacian matrix $\mathbf{L}$ as the Fourier basis. Instead, our model is a relational feature learning model, so we use the eigenvectors of the transition probability matrix $\mathbf{P}$ as the Fourier basis; 3) Since we aim at learning the relational features, our convolutional filters donot need the attributes of the node itself, i.e., do not need to consider the case of $k = 0$ in Eq. (5); 4) The convolution output of our model is relational features, and most conventional models are content features; 5) Our model can deal with HINs, while traditional graph convolution models cannot work on HINs.

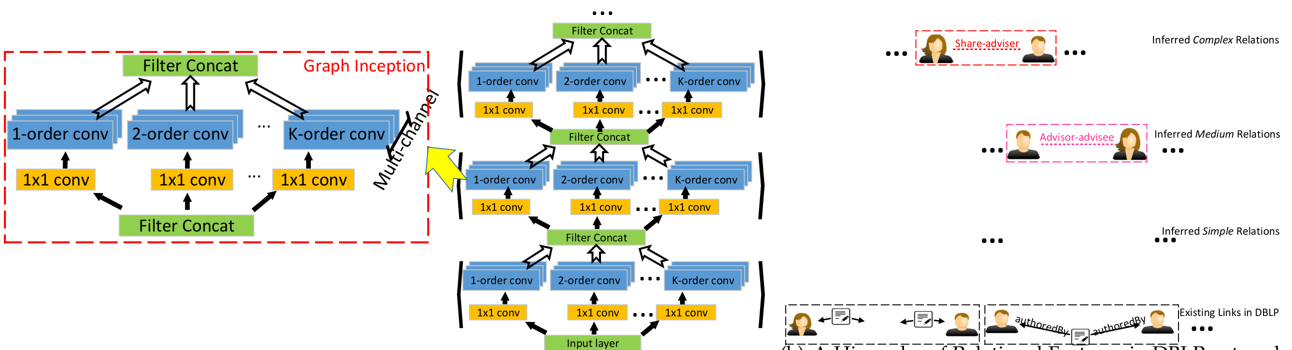
4 Graph Inception Module-based Deep Relational Feature Learning

In order to balance the complexity levels of relational features, we proposed the *graph inception module*, to automatically generate a hierarchy of relational features from simple to complex features.

Conventional inception modules can only work on euclidean grids data, e.g., image data [14], and can not be used on general graph structure, e.g., network data. Therefore, we propose the *graph inception module*, in order to learn the relational features in networks more effectively. As shown in Fig. 6a, graph inception module combines convolutional kernels with different size to extract relational features, and generate a hierarchy of relational features from simple to complex features through stacking such network layers. We also give a toy example in Fig. 6b, which generate relationships among authors in the DBLP network, from simple to complex ones. For instance, we take two convolutional kernels for each channel on every layer, and set the kernel sizes as 1 and 2 respectively. Then the graph inception module in the tth layer is defined as:

$$\begin{aligned} \mathbf{C}_{\ell 1}^t &= \mathbf{P}_\ell\sigma(\hat{\mathbf{H}}^{t-1})\Theta_{\ell 1}^t \\ \mathbf{C}_{\ell 2}^t &= \mathbf{P}_\ell\sigma(\hat{\mathbf{H}}^{t-1})\Theta_{\ell 1'}^t + \mathbf{P}_\ell^2\sigma(\hat{\mathbf{H}}^{t-1})\Theta_{\ell 2}^t \\ \hat{\mathbf{H}}^t &= r\left(\mathbf{C}_{11}^t, \mathbf{C}_{12}^t, \ldots, \mathbf{C}_{|\mathcal{S}|1}^t, \mathbf{C}_{|\mathcal{S}|2}^t\right). \end{aligned} \tag{8}$$

Here $\mathbf{C}_{\ell 1}^t/\mathbf{C}_{\ell 2}^t$ are the convolution kernels of size 1/2, by combining them to construct the tth layer of the neural



(a) Framework of the graph inception module (b) A Hierarchy of Relational Features in DBLP network

Fig. 6. Graph inception module. (a) The framework of the proposed module, and the red chart on the left is an enlarged view of each layer. (b) A toy example of the graph inception module in DBLP network, which generate a hierarchy of relationships among authors via graph inception (The green arrow indicates the neural network in the left parentheses).

network. And $\mathbf{C}_{\ell 1}^t, \mathbf{C}_{\ell 2}^t \in \mathbb{R}^{n\times F}$. $\sigma(\cdot)$ is a 1×1 convolutional filter for dimensionality reduction, and $\hat{\mathbf{H}}^0 = (X) \in \mathbb{R}^{1\times n\times C}$. When t is 0, $\mathbf{\Theta}_{\ell k}^t \in \mathbb{R}^{C\times F}$ $(k=1,2)$, otherwise $\mathbf{\Theta}_{\ell k}^t \in \mathbb{R}^{F\times F}$. To further reduce the number of parameters, we replace $C_{\ell 2}^t$ as:

$$\mathbf{C}_{\ell 2}^t = \mathbf{P}_\ell^2 \sigma(\hat{\mathbf{H}}^{t-1})\mathbf{\Theta}_{\ell 2}^t. \tag{9}$$

We can extracting complex relational features through stacking multiple graph inception layers, and control the complexity of the learned relational features by adjusting the number of layers.

Compared with Eq. (7), the graph inception module-based method has three strengths: 1) It can significantly reduce the required storage space when K is a large number (Eq. (7) need to store K matrices while Eq. (8) only need to store $\mathbf{P}$ and $\mathbf{P}^2$); 2) It requires fewer parameters than Eq. (7) when the network is very deep; 3) It can enhance the ability of the model to extract the relational features. We intuitively expect that such a model can alleviate the problem of overfitting on local neighborhood structures for graphs with very wide node degree distributions, such as social networks, citation networks, etc.

Suppose the graph inception module has T layers, then the convolution operation has complexity $\mathcal{O}(|\mathcal{S}|\cdot C\cdot F\cdot T\cdot|\mathcal{E}|)$, here $|\mathcal{S}|$ is the number of the meta paths, C is the dimension of the input signals, and F is the hidden dimension of convolutional filters. As $\mathbf{P}_\ell^k\mathbf{X}$ can be efficiently implemented as a product of a sparse matrix with a dense matrix, and $|\mathcal{S}|, C, F, T \ll |\mathcal{E}|$, the complexity is linear in the number of graph edges.

5 Final Solution

After learning the relational features for all nodes, we predict the label $Y_i \in \mathcal{Y}_\mathcal{U}$ via a softmax layer with relational features $\mathbf{H}_i^T$ and local features $\mathbf{x}_i$, i.e.,

$$Pr(Y_i = c|\mathcal{Y}_{\mathcal{N}_i}, \mathbf{x}_i) = \text{softmax}(\mathbf{vec}(W_{c\mathbf{H}}\mathbf{H}_i^T) + W_{c\mathbf{x}}\mathbf{x}_i + \mathbf{b}_c), \tag{10}$$

where $W_{c\mathbf{H}} \in \mathbb{R}^{2|\mathcal{S}|\times F}$ and $W_{c\mathbf{x}} \in \mathbb{R}^d$ are weight matrices and $\mathbf{b}_c$ is a bias. T is the top layer of Eq. (8), and $c \in \{1,\ldots,C\}$ indicates the label of nodes. Here $\mathbf{vec}(\cdot)$ is a function which scales the input matrix into a vector. We use the labels of all nodes as the input signal $\mathbf{X}$ of Eq. (8), and use $\mathbf{0}$ to initialize the labels of the test (unknown) nodes. There are two reasons why we only use the labels instead of both labels and local features as the input signal $\mathbf{X}$: 1) It can significantly reduce the number of parameters; 2) Conventional collective classification methods confirmed that there is only little association between the labels of the node and the local features of the neighboring nodes.

In this section, we propose two versions of the models with different inference styles based on Eq. (10), to solve the collective classification problem in HINs, both are called GraphInception algorithm. One is based upon iterative inference (called ICA-based GraphInception), the other is based upon stacked inference (called STACK-based GraphInception). The architecture of the proposed GraphInception is showed in Fig. 5.

5.1 ICA-Based GraphInception Algorithm

Inspired by the success of iterative classification method [1], we first propose an ICA-based collective classification algorithm in HINs. The algorithm of ICA-based GraphInception is shown in Fig. 7, which includes the following steps:

Multi-Channel Network Construction.Given a HIN G, we first extract a meta-path set $\mathcal{S} = \{\mathcal{P}_1,\ldots,\mathcal{P}_{|\mathcal{S}|}\}$ within the maximum path length $p_{\max}$. Then we use the meta-path set to construct the multi-channel network G'.

Training Model. In the training step, we construct an extended training set $\mathcal{D} = \{(\mathbf{x}_i', Y_i)\}$ by converting each instance $\mathbf{x}_i$ to $\mathbf{x}_i' = (\mathbf{vec}(\mathbf{H}_i^T), \mathbf{x}_i)$ using local fecatures $\mathbf{x}_i$ and relational features learned from Eq. (8). Then we train the neural network on the extended training set based on Eq. (10).

Iterative Inference. In the inference step, we iteratively update the label values of neighboring nodes based on latest predict results, and then use these new labels to make prediction. The iterative process terminates when the convergence criteria are met. In the end, we will get $\mathcal{Y}_\mathcal{U}$ for the test instances.

5.2 STACK-Based GraphInception Algorithm

As discussed previously, the iterative inference process may not be highly efficient during inference. In order to improve the efficiency of the collective inference process, we present the STACK-based GraphInception algorithm for collective classification problem in HINs, which inspired by the stacked learning algorithm [21]. The framework of STACK-based GraphInception is shown in Fig. 8. The algorithm includes the following steps:

```
Input:
  $G$ : a heterogeneous network,    $T$: depth of inception layer    $\mathcal{X}$: features for all instances
  $Max_{It}$: maximum # of iterations,  $p_{max}$: maximum metapath length (default=4)
Multi-channel Network Construction:
  - Construct the meta-path set $\mathcal{S} = \{\mathcal{P}_1, \cdots, \mathcal{P}_{|\mathcal{S}|}\}$
    Breadth search on schema graph of $G$, starting from $\mathcal{V}_1$ by adding meta-path $\mathcal{P}_\ell$ that ends with $\mathcal{V}_1$ into $S$, $\mathbf{length}(\mathcal{P}_\ell) \leqslant p_{max}$.
  - Use the meta path set $\mathcal{S}$ to construct the multi-channel network $G'$ of the HIN $G$.
Initialization:
  - For each $v_{1i} \in \mathcal{U}$, set the label $Y_i$ as $Y_i = \mathbf{0}$
Training:
  - Learn the neural network model $f$:
    1. Construct an extended training set $\mathcal{D} = \{(\mathbf{x}'_i, Y_i)\}$ by converting each instance $\mathbf{x}_i$ to $\mathbf{x}'_i$ with $T$ layers of Eq. 8
    2. Let Eq. 10 be the neural network $f$ trained on $\mathcal{D}$.
Iterative Inference:
  - For each $v_{1i} \in \mathcal{U}$, estimate the label $Y_i$ as $Y_i = f((\mathbf{x}_i, \mathbf{0}))$
  - Repeat until convergence or #iteration $> Max_{It}$
    1. Construct the associate features $\mathbf{x}'_i$ for each test instance $v_{1i} \in \mathcal{U}$ as: $\mathbf{x}'_i = \big(\mathbf{x}_i, \mathbf{vec}(\mathbf{H}_i^T)\big)$, where $\mathbf{H}_i^T$ is estimated through Eq. 8
    2. Update the $Y_i$ as $Y_i = f(\mathbf{x}'_i)$ for each $v_{1i} \in \mathcal{U}$.
Output:
  $\mathcal{Y}_\mathcal{U} = \big(Y_1, \cdots, Y_{|\mathcal{U}|}\big)$:   the labels of test instances ($v_{1i} \in \mathcal{U}$).
```

Fig. 7. The ICA-based GraphInception algorithm.

Multi-Channel Network Construction. The same as the previous algorithm. Given a HIN G, we first extract a meta-path set $\mathcal{S} = \{\mathcal{P}_1, \ldots, \mathcal{P}_{|\mathcal{S}|}\}$ within the maximum path length $p_{\max}$. Then we use the meta-path set to construct the multi-channel network G'.

Stacked Training. In this step, we train a neural network model, which includes two steps: First, we construct the initialization layer to predict the nodes labels based on their local features. Then, we stacked the Eq. (10) as the hidden layer and the final layer, to combine the relational features and the local features. For each layer, we convert each instance $\mathbf{x}_i$ to $\mathbf{x}'_i = (\mathbf{vec}(\mathbf{H}_i^t), \mathbf{x}_i)$ using local fecatures $\mathbf{x}_i$ and relational features learned from Eq. (7). In particular, we use Relu function to replace the softmax layer in the hidden layer. Finally, we train the whole neural network with the training set.

Stacked Inference. In the inference step, we just use the local features of the test set as input, and use the trained neural network model to predict the labels. Then we will get $\mathcal{Y}_\mathcal{U}$ for each test instance.

Compared with ICA-based GraphInception algorithm, the STACK-based GraphInception algorithm is more efficient in the inference step. Moreover, [22] had proposed some innovative ideas for stacked learning in neural network, such as parameter sharing, highway network, which can be easily integrated into our STACK-based GraphInception algorithm. However, the stack-based GraphInception usually needs to learn more parameters than ICA-based GraphInception. Therefore, which algorithm to choose, being depending on the circumstances.

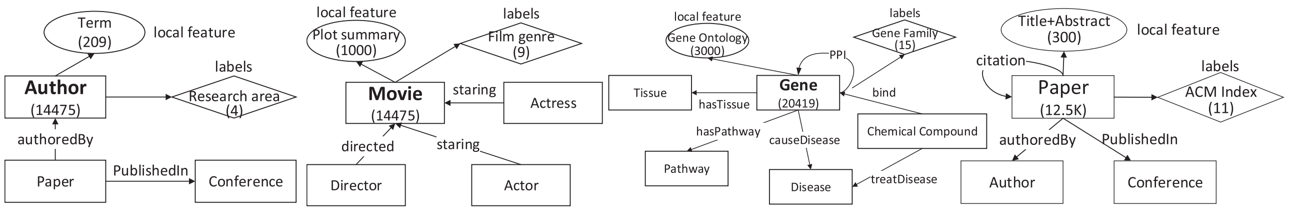
6 Experiments

6.1 Data Collection

In order to validate the collective classification performances, we apply our algorithm to four real-world HINs. Note

```
Input:
  $G$ : a heterogeneous network,    $T$: depth of inception layer    $\mathcal{X}$: features for all instances
  $Depth_S$: depth of stack layer    $p_{max}$: maximum metapath length (default=4)
Multi-channel Network Construction:
  - Construct the meta-path set $\mathcal{S} = \{\mathcal{P}_1, \cdots, \mathcal{P}_{|\mathcal{S}|}\}$
    Breadth search on schema graph of $G$, starting from $\mathcal{V}_1$ by adding meta-path $\mathcal{P}_\ell$ that ends with $\mathcal{V}_1$ into $S$, $\mathbf{length}(\mathcal{P}_\ell) \leqslant p_{max}$.
  - Use the meta path set $\mathcal{S}$ to construct the multi-channel network $G'$ of the HIN $G$.
Stacked Training:
  - Learn the neural network model $f$:
    1. Construct the initialization layer as the first layer of $f$, i.e., $f^0(\mathbf{x})$, which only use the local features of the instances to predict their labels.
    2. For $t$ in 1 to $Depth_S$:
       - Converting each instance $\mathbf{x}_i$ to $\mathbf{x}'_i = (\mathbf{vec}(\mathbf{H}_i^t), \mathbf{x}_i)$ using local fecatures $\mathbf{x}_i$ and relational features learned from Eq. 7.
       For relational features, the input $\mathbf{X}$ of Eq. 7 is the predict label of the previous layer, i.e., $\mathbf{X} = f^{t-1}(\mathbf{x}')$. In particular, $\mathbf{X} = f^t(\mathbf{x})$ when $t = 0$.
       - Stacked the hidden layer $f^t$ with Eq. 10, i.e., $f^t(\mathbf{x}')$.
    3. Training the neural network $f$ with training set.
Inference:
  - For each $v_{1i} \in \mathcal{U}$, estimate the label $Y_i$ as $Y_i = f(\mathbf{x}_i)$.
Output:
  $\mathcal{Y}_\mathcal{U} = \big(Y_1, \cdots, Y_{|\mathcal{U}|}\big)$:   the labels of test instances ($v_{1i} \in \mathcal{U}$).
```

Fig. 8. The STACK-based GraphInception algorithm.



(a) DBLP Dataset (b) IMDB Dataset (c) SLAP Dataset (d) ACM Conference Dataset

Fig. 9. Schema of datasets.

that the IMDB dataset is a multi-label dataset, and the rest are multi-class datasets.

- The first dataset, i.e., DBLP four areas [23], is a bibliographic information network extracted from DBLP[1], which involves three types of nodes: conference, paper and author, connected by two types of relations/links: authoredBy link and publishedIn link. We treat authors as our target instances, with the research area of the authors as the instances labels. We also extract a bag-of-words representation of all the paper titles published by the author as local features, which include 209 words (terms). For detailed description of the DBLP dataset, please refer to [23], and the network schema is shown in Fig. 9a.
- *IMDB Dataset:* The second dataset is a movieLens dataset,[2] which contains four types of nodes: movie, director, actor and actress, connected by two types of relations/links: directed link and actor/actress staring link. The target instance type is movie instance, which assigned with a set of class labels, indicating genres of the movie. For each movie, we also extract a bag-of-words vector of all the plot summary about the movie as local features, which include 1000 words. The network schema of IMDB dataset is shown in Fig. 9b. For detailed description of the IMDB dataset, please refer to [22].
- *SLAP Dataset:* The third dataset is a bioinformatic dataset SLAP [24]. As showed in Fig. 9c, the SLAP dataset contains integrated data related to chemical compound, gene, disease, tissue, pathway etc. We treat genes as our target instances. Each gene can belong to one of the gene family. We extract 15 most frequent gene families as the instances labels, and extract 3000 gene ontology terms (GO terms) as the features of each gene instance. For detailed description of the SLAP dataset, please refer to [10].
- *ACM Conference Dataset:* The last dataset is also a bibliographic information network, ACM Conference dataset,[3] and the network schema is shown in Fig. 9d. This network includes 196 conference proceedings (e.g., KDD'10, KDD'09, etc.), 12.5K papers and 17K authors. On each paper node, we extract a bag-of-words representation of the paper title and abstract to use as local features, which include 300 words. Each paper node in the network is assigned with a class label, indicating the ACM index term of the paper including 11 categories. For detailed description of the ACM Conference dataset, please refer to [10].

6.2 Compared Algorithms

To demonstrate the effectiveness of our method, we compare with the following state-of-the-art algorithms (summarized in Table 3):

- Logistic Regression (LR for short): baseline algorithm.
- Highway Network [13]: A type of neural network layer that uses a gating mechanism to control the information flow through a layer. Stacking multiple highway layers allow for training of deep networks. We share parameters for each layer.
- ICA [1]: A basic collective classification method in homogeneous networks.
- HCC [10]: This is a collective classification approach, which works on HIN by exploiting dependencies based on multiple meta paths in the network.
- LinBP [25]: This is a linearization of belief propagation method for homogeneous networks, which guarantees exact convergence.
- Stacked Learning [26]: This is a multi-step learning procedure for collective classification. In each step, the predict labels of the neighbor nodes and the local features of the target node are fed into a standard classifier (LR) to make predictions.
- Column Network (CLN for short) [22]: This is a deep feedforward network for collective classification in multi-relational domains, with shared parameters for each layer. It also implements the highway layer into the model.

TABLE 3
Types of Models, Based on the Kinds of Features Used

Method	Self attr.	Neigh. labels	Deep nets	HINs	Publication and year
LR	✓				—
Highway Network	✓		✓		Srivastava. 2015
ICA	✓	✓			Sen. 2008
HCC	✓	✓		✓	Kong. 2012
LinBP		✓			Gatterbauer. 2015
Stacked Learning	✓	✓		✓	Choetkiertikul. 2015
CLN	✓	✓	✓	✓	Pham. 2017
GCN	✓	✓	✓		Kipf. 2017
GCN (metapath)	✓	✓	✓	✓	This paper
HAN	✓	✓	✓	✓	Wang. 2019
GraphInception	✓	✓	✓	✓	This paper

1. http://dblp.uni-trier.de/db/
2. http://www.imdb.com
3. http://dl.acm.org/

- GCN [27]: This is a graph convolution-based semi-supervised classification algorithm. The model extracts the 1-localized information for each node in each convolution layer, and extracts the deeper relational features through stacked multiple convolution layers. However, it focuses on homogeneous networks.
- GCN (metapath): In order to compare with the GCN method more fairly, we extend the GCN method into HINs with meta path, as described in section 3.1.
- HAN [28]: This is a heterogeneous graph neural network based on the hierarchical attention, including node-level and semantic-level attentions. The model aggregates features from meta-path based neighbors.
- GraphInception: This is our method proposed in Section 5.

In practice, we make use of Keras for an efficient GPU-based implementation of all algorithms. For a fair comparison, we train all models for 1500 epochs (training iterations) using RMSprop with a learning rate of 0.01. All neural nets use ReLU in the hidden layers. The maxmum inference iteration MAX_{It} is 10, we also search for the number of stacked layers for stacked-based methods (CLN, stacked learning, highway network): 5,10,15,20. The maximum of the meta-path length p_{max} is 4. The hidden dimension of the convolutionl filters is 4 times the number of label types, i.e., $F = 4 * C$. Each inception module has two convolutional filters, and the kernel size K is 1 and 2 respectively. For hyper-parameter tuning, we search for the number of inception layers T: 1,2,3,4. We set the belief matrix of LinBP as the transition matrix minus the inverse of the number of labels elementwisely [25]. That is, the belief matrix is not learned by a neural network in LinBP, so the stacking method will not work. For heterogeneous network, we vote on the results of each channel. Note that, if the belief matrix is not default, LinBP can be regarded as ICA. 5-fold cross validation is used in all experiments, and the results are reported by the mean results of 10 runs. In the following experiments, the GraphInception algorithm refers to the ICA-based GraphInception unless otherwise specified. In Section 6.5, we will discuss the effectiveness and the efficiency of the two versions of GraphInception. Code for our model can be found on Github.[4]

6.3 Multi-Class Classification Performance

In our first experiment, we evaluate the effectiveness of the proposed GraphInception method on three multi-class classification problem: predicting the research area for authors in DBLP network, predicting the gene family for genes in SLAP network, and predicting the ACM index for papers in ACM Conference network. The evaluation metrics include accuracy and F1-score. The results are reported in Table 4. Performance ranks of each model on each of the evaluation criteria are also listed.

On DBLP dataset, GraphInception works best with 4 inception layers. On SLAP dataset, GraphInception works best with 1 inception layers. On ACM Conference dataset, GraphInception works best with 4 inception layers. The first observation in the Table 4 is: Almost all have better performance than the baseline logistic regression, which demonstrates that both deep learning models and collective classification models can improve classification performance. We also find that HCC, Stacked learning are significantly outperforms than Highway network and ICA method. These results support that the heterogeneous dependencies among instances can improve classification performance. Compared to GraphInception, though HCC also utilizes multi-channels for classification, it only considers first-order neighbors. HCC is actually a special case of our model. When the inception size of kernels of GraphInception is set to 1 and the depth of the network of GraphInception is set to 1, our GraphInception is HCC. So the results of HCC here are equivalent to the results of our model where parameters are set to extreme values. Because HAN does not explicitly utilize relational features, the results of our model are better on most data sets. The essence of GCN is the smoothing of attributes, so when the attributes are low-dimensional and dense, it would cause the over-smoothing problem, which results in the poor performance of GCN [29]. Although the CLN and GCN methods can capture the deep relational features in networks which use the local features as input, but they don't perform well on some datasets. One possible reason is that the node labels are more closely related to the labels of the neighboring nodes, rather than the features of the neighboring nodes. Compared with all above algorithms, GraphInception always has best performance in the multi-class classification task.

TABLE 4
Results on the DBLP, SLAP and ACM Conference Datasets

Method	Self attr.	Neigh. labels	Deep nets	HINs	Publication &year
LR	✓				—
Highway Network	✓		✓		Srivastava. 2015
ICA	✓	✓			Sen. 2008
HCC	✓	✓		✓	Kong. 2012
LinBP		✓			Gatterbauer. 2015
Stacked Learning	✓	✓		✓	Choetkiertikul. 2015
CLN	✓	✓	✓	✓	Pham. 2017
GCN	✓	✓	✓		Kipf. 2017
GCN (metapath)	✓	✓	✓	✓	This paper
HAN	✓	✓	✓	✓	Wang. 2019
GraphInception	✓	✓	✓	✓	This paper

"↓" indicates the smaller the value the better performance; "↑" indicates the larger the value the better performance.

6.4 Multi-Label Classification Performance

In our second experiment, we evaluate the effectiveness of the proposed GraphInception method on a multi-label classification problem: predicting the film genres for movies in IMDB network. The evaluation metrics include: hamming loss, micro F1-score and subset 0/1 loss. The results are reported in Table 5. Performance ranks of each model on each of the evaluation criteria are also listed.

On IMDB dataset, GraphInception works best with 1 inception layers. The first observation in Table 5 we have is that most collective classification methods have better performance than the methods which do not consider the correlations among instances, i.e., LR and highway network. We also find that HCC, Stacked learning and GCN

4. https://github.com/zyz282994112/GraphInception.git

TABLE 5
Results (Mean (Rank)) on the IMDB Dataset

Algorithms	Evaluation Criteria			Avg. Rank
	Hamming loss↓	Micro F1↑	Subset 0/1 loss ↓	
GraphInception	**0.227** (1)	**0.551** (1)	**0.879** (1)	1
Stacked learning	0.251 (3)	0.533 (3)	0.901 (2)	2.67
GCN (metapath)	0.242 (2)	0.512 (5)	0.903 (3)	3.33
HCC	0.289 (5)	0.550 (2)	0.945 (4)	3.67
ICA	0.317 (6)	0.524 (4)	0.957 (6)	5.33
HAN	0.261 (4)	0.410 (10)	0.961 (7)	7
LR	0.341 (7)	0.503 (6)	0.967 (8)	7
CLN	0.406 (9)	0.417 (9)	0.954 (5)	7.67
GCN	0.388 (8)	0.479 (7)	0.968 (9)	8
Highway network	0.481 (10)	0.435 (8)	0.994 (10)	9.33

"↓" indicates the smaller the value the better performance; "↑" indicates the larger the value the better performance.

(with metapath) are significantly outperform than GCN and ICA method. These results support that the heterogeneous dependencies among instances can improve classification performance. Overall, GraphInception has best performance on all metrics in the multi-label classification task.

6.5 Evaluation of Efficiency Performance

In our third experiment, we compared the two GraphInception algorithms (ICA-based and STACK-based) in terms of efficiency and effectiveness. The evaluation metrics of the efficiency performance is the average inference time of 10 epoch measured in microseconds wall-clock time. The efficiency performance results are reported in Fig. 10, including two datasets (DBLP and IMDB). We also compared the effectiveness of both algorithms, and the results are showed in Fig. 11.

In Fig. 10, we can find that the efficiency of the STACK-based GraphInception is significantly higher than ICA-based GraphInception in almost datasets, since the STACK-based GraphInception does not need to iterative inference the labels of the test nodes. We also find that ICA-based GraphInception and stack-based GraphInception have similar effective performance in all datasets as shown in Fig. 11, both algorithms can effectively solve the collective classification problem in HINs. Overall, these results demostrate that the STACK-based GraphInception algorithm is more efficient than ICA-based GraphInception in inference step.

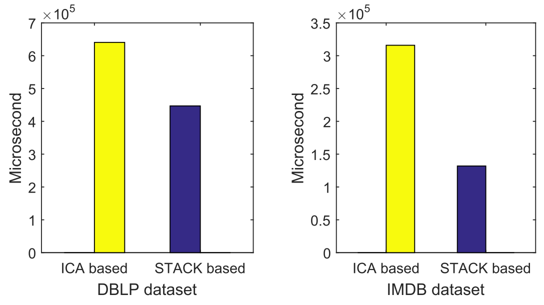


Fig. 10. The efficiency performance of the two versions of GraphInception.

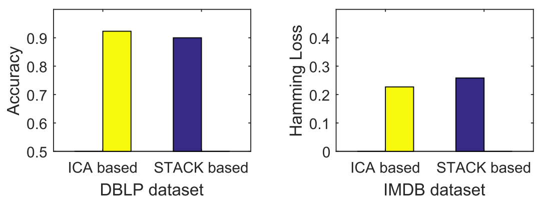


Fig. 11. The effectiveness performance of the two versions of GraphInception.

6.6 Relational Features Visualization

To better understand the learned relational features of our model, we use $t-SNE$ [30] to visualize the hidden layer activations of Eq. (8), trained on DBLP dataset. We also compare GraphInception with other deep learning models, including: highway network, stacked learning and CLN. The visualization results are shown in Fig. 12, colors represent research area class of authors. We can find that our GraphInception model can effectively gather the same type nodes together (same color). The results demonstrate the effectiveness of the proposed Eq. (8) on learning the deep relational features. We do not compare GraphInception with GCN which uses both local features and neighbor nodes labels in hidden layer because it is unfair to other models (including our model) which only use neighbor nodes labels in hidden layer.

In order to demonstrate the effectiveness of the proposed *multi-channel network*, we also visualize the hidden layer activations of different homogeneous networks and of our multi-channel network on DBLP network. The results are shown in Fig. 13.

In Fig. 13, first we can find that with the increase of epochs, the hidden relation can be better expressed and multi-channel network has better visualizes performance than homogeneous network visualize performance on most epochs. It demonstrates our claim that HIN can help to learn the complex dependencies than simple homogeneous networks. Second, the visualization results of multi-channel network looks like a fusion of co-author network and co-conference network. Because the multi-channel network can be regarded as a combination of multiple homogeneous networks in different channels.

Overall, we can find the proposed GraphInception model can effectively learn the relational features in networks through those visualization results.

6.7 Evaluation of Convolution Graph

We compare different convolution graph of our proposed GraphInception model on DBLP network and IMDB network, and the results are reported in Fig. 14. The convolution graph of our original GraphInception model is transition probability matrix $\mathbf{P}$ (red line). In all other cases, the convolution graph in Eq. (8) is replaced by adjacency matrix $\mathbf{A}$, laplacian matrix $\mathbf{L}$ and $\mathbf{GCN} = \hat{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\hat{\mathbf{D}}^{-\frac{1}{2}}$

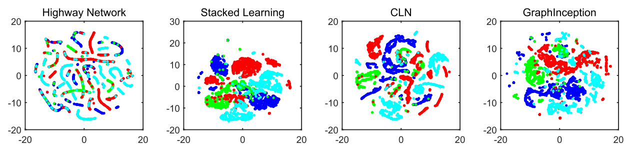


Fig. 12. Visualize the hidden layer activations on DBLP dataset.

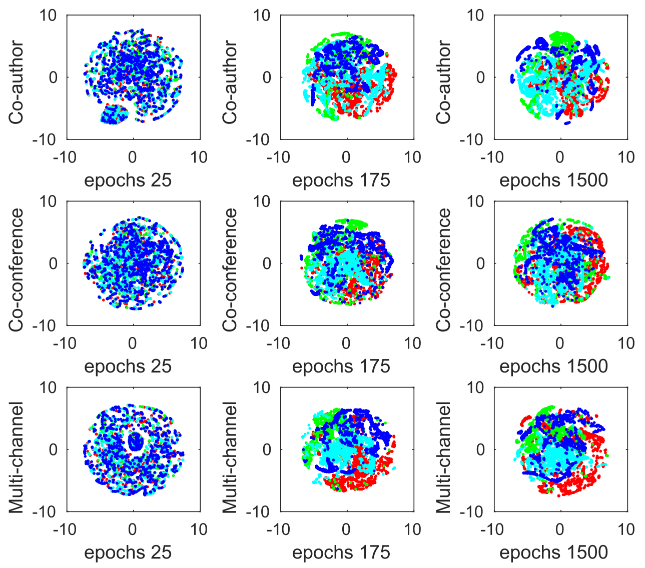


Fig. 13. Visualize the hidden layer activations on DBLP dataset with different networks (only co-author network, only co-conference network and multi-channel network).

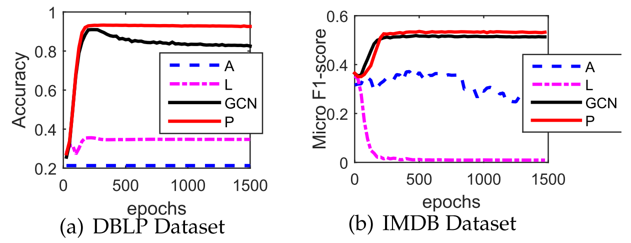


Fig. 14. Comparison of different convolution graphs.

in [27], $\hat{\mathbf{D}}_{ii} = \sum_j (\mathbf{A}+\mathbf{I})_{ij}$. We can find that $\mathbf{A}$ does not perform well on both dataset, one possible reason is that it doesn't normalise the convolution result in each layer. We also find that GraphInception can get best performance with $\mathbf{P}$, which supports our intuition that graph transition probability matrix can effectively extract the relational features in networks. Similar trend holds for the other datasets which cannot be shown due to space limitation.

6.8 Evaluation of Different Input

In this experiment, we validate GraphInception on different input X of Eq. (8), including: labels, local features and labels + local features. The results are reported in Fig. 15, including two datasets (DBLP and IMDB). Similar trend holds for the other datasets which cannot be shown due to space limitation. We can find that GraphInception with labels always has best performance on every dataset, and GraphInception with both labels and local features has sub-optimal performance, while GraphInception with local features has worst performance on each dataset. These results support our claim in Section 5 that the node label associated with the neighboring nodes labels, regardless of the neighboring nodes features.

6.9 Parameters Sensitivity

There exist three essential hyper-parameters in GraphInception: the convolutional kernel size K, the hidden dimension of convolutional filters F, and the maximum number of iterations Max_{Iter} in the iterative inference part of ICA-based

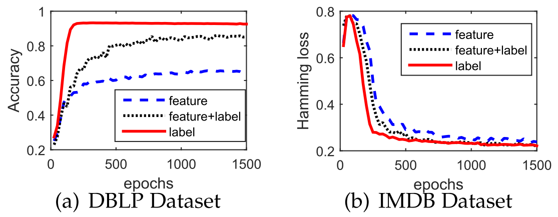


Fig. 15. Comparison of different input $\mathbf{X}$.

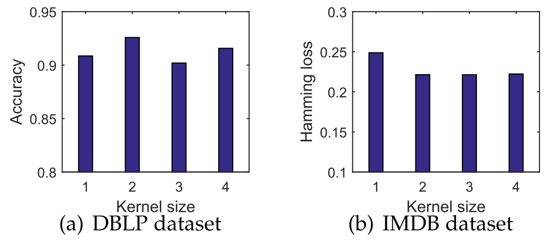


Fig. 16. Performances of different kernel sizes.

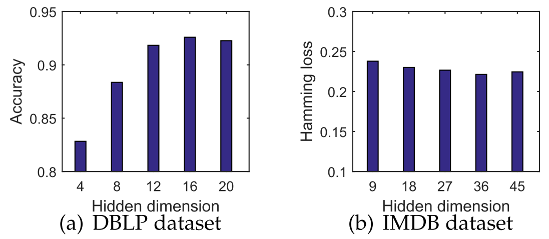


Fig. 17. Performances of different hidden dimension.

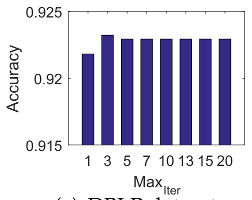
GraphInception. To test the stability of the performances of GraphInception method, we test different values of K, F and Max_{Iter} on both DBLP and IMDB dataset. Similar trend holds for the other two datasets which cannot be shown due to space limitation. The results are shown in Figs. 16, 17, and 18 .

In Fig. 16, we can find that it is not sensitive to the kernel sizes on both DBLP dataset and IMDB dataset. In Fig. 17, we can find that more hidden filters can achieve better performance on both DBLP dataset and IMDB dataset. In Fig. 18, when the number of iterations is very small, e.g., $Max_{Iter} < 5$, the results did not perform well, because the algorithm did not reach convergence yet. However, with the increase of the number of iterations, the results do not change obviously.

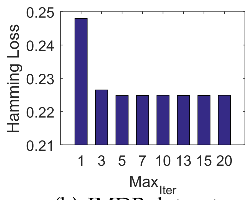
7 Related Work

This paper sits at the intersection of two developed areas: collective classification and deep learning models. We provide a brief overview of related works in both fields.

Collective classification [1], [4], [6], [31], [32] of relational data, has been investigated by many researchers. Basic collective classification problems mainly focus on homogeneous network [5], [25], [33]. And usually, these models assume that the nodes belonging to the same class tend to link to each other. Graph walk-based methods, such as PageRank [34], personalized PageRank [35], and random walk with restart (RWR) [36], and belief propagation-based methods, such as LinBP [25] are the common algorithms. In fact, both methods can be regarded as special ICA which estimate parameters in a specific method.



(a) DBLP dataset



(b) IMDB dataset

Fig. 18. Performances of different Max_{Iter}.

Graph walk-based methods and belief propagation-based methods use adjacency matrices or both adjacency matrices and belief propagation matrices as parameters, while ICA can learn parameters by gradient descent, which fits the data better. Ji [37] studied a specialized classification problem on HINs, where different types of nodes share a same set of label concepts. Kong et al. [10], [11], [38] proposed methods based on meta-path to solve the collective classification problem on one-type nodes in HINs. The HCC [10] proposed by Kong is actually a special case of our model whose size of inception kernels is 1 and depth of the network is 1. Kou [39] proposed a method based on stacked model to solve collective classification problem. Choetkiertikul [26] extends the stacked model into multi-relational networks, which can be considered as one of multi-channel networks proposed in this paper. However, all above are shallow models. Nandanwar [40] proposed a deep random walk-based collective classification model, but it focuses on homogeneous networks. In this work, we chose three representative algorithms in [1], [10], [40] as competing algorithms.

On the other hand, in deep learning models, there exist many related works on graph convolution recently [12], [18], [41], [42], [43]. There are two strategies to define convolutional filters on graph: either from a spatial approach or from a spectral approach. By construction, spatial approaches provide filter localization via the finite size of the kernel. However, although graph convolution directly in the spatial domain is conceivable, it also faces the challenge of matching local neighborhoods, as pointed out in [12]. On the other hand, the convolution theorem defines convolutions as linear operators that diagonalize in the Fourier basis (usually represented by the eigenvectors of the Laplacian operator), which provide a well-defined translation operator on graphs. However, it is difficult to represent the local neighborhoods in the spectral domain. Recent works [12], [42] tried some localized filters to overcome the problem, which inspired our algorithm. There also exist other deep models which are applied on collective classification problem. Wang [44] proposed a deep network embedding method, which can be used to solve the classification problem on networks. Kipf [27] proposed a semi-supervised classification method based on graph convolution, while Moore [45] proposed a semi-supervised classification method based on RNN model. Rossi [46] proposed a model based on feature diffusion methods which can catch the features from across-networks for transfer learning tasks. However, all of them focused on homogeneous networks. Pham [22] proposed a collective classification method on multi-relational networks based on stacked model [39] and highway networks [13]. Wang [28] proposed a collective classification method on heterogeneous information networks based on graph attention networks. In this work, we also choose three representative algorithms in [13], [22], [25], [27], [28] as competing algorithms.

8 Conclusion

In this paper, we proposed a graph convolution-based model for learning the deep relational features in HINs, which mainly focus on collective classification problem. We further proposed the *graph inception module* to mix both complex and simple dependencies among the instances. Empirical studies on real-world tasks demonstrate the effectiveness of the proposed GraphInception algorithm in learning deep relational features in HINs.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China Projects No.U1636207, No.91546105, the Shanghai Science and Technology Development Fund No.19511121204, No.19DZ1200802, No. 16JC1400801.

References

[1] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–106, 2008.

[2] W. Gatterbauer, "Semi-supervised learning with heterophily," *CoRR*, vol. abs/1412.3100, 2014. [Online]. Available: http://arxiv.org/abs/1412.3100

[3] R. A. Rossi, R. Zhou, N. K. Ahmed, and H. Eldardiry, "Relational similarity machines (RSM): A similarity-based learning framework for graphs," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 1807–1816.

[4] J. Neville and D. Jensen, "Iterative classification in relational data," in *Proc. Workshop Statistical Relational Learn., 17th Nat. Conf. Artif. Intell.*, 2000, pp. 42–49.

[5] L. K. Mcdowell and D. W. Aha, "Labels or attributes?: Rethinking the neighbors for collective classification in sparsely-labeled networks," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, 2013, pp. 847–852.

[6] I. Alodah and J. Neville, "Combining gradient boosting machines with collective inference to predict continuous values," *CoRR*, vol. abs/1607.00110, 2016. [Online]. Available: http://arxiv.org/abs/1607.00110

[7] Y. Lecun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] A. Coates, "Deep learning for machine vision," in *Proc. Brit. Mach. Vis. Conf.*, Bristol, UK, Sept. 9–13, 2013, p. 1.1, doi: 10.5244/C.27.1.

[9] R. Socher, Y. Bengio, and C. D. Manning, "Deep learning for NLP (without magic)," in *Proc. 50th Annu. Meet. Assoc. Comput. Linguistics*, 2012, Art. no. 5.

[10] X. Kong, P. S. Yu, Y. Ding, and D. J. Wild, "Meta path-based collective classification in heterogeneous information networks," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 1567–1571.

[11] Y. Zhang, Y. Xiong, X. Kong, and Y. Zhu, "Netcycle: Collective evolution inference in heterogeneous information networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1365–1374.

[12] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. 30th Advances Neural Inf. Process. Syst.*, 2016, pp. 3837–3845.

[13] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *CoRR*, vol. abs/1505.00387, 2015. [Online]. Available: http://arxiv.org/abs/1505.00387

[14] C. Szegedy et al., "Going deeper with convolutions," in *Proc. 28th IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[15] Y. Sun and J. Han, *Mining Heterogeneous Information Networks: Principles and Methodologies*. San Rafael, CA, USA: Morgan & Claypool, 2012.

[16] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *TiiS*, vol. 5, no. 4, pp. 19:1–19:19, 2016.

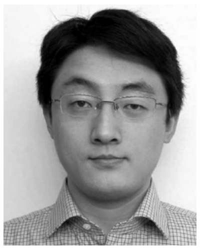
[17] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang, "Discovering meta-paths in large heterogeneous information networks," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 754–764.

[18] M. Henaff, J. Bruna, and Y. Lecun, "Deep convolutional networks on graph-structured data," *CoRR*, vol. abs/1506.05163, 2015.

[19] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed. Cambridge, MA, USA: Academic Press, 1999.
[20] N. Guberman, "On complex valued convolutional neural networks," *CoRR*, vol. abs/1602.09046, 2016. [Online]. Available: http://arxiv.org/abs/1602.09046
[21] A. S. Fast and D. D. Jensen, "Why stacked models perform effective collective classification," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 785–790.
[22] T. Pham, T. Tran, D. Q. Phung, and S. Venkatesh, "Column networks for collective classification," in *Proc. Workshop Statistical Relational Learn., 31st Nat. Conf. Artif. Intell.*, 2017, pp. 2485–2491.
[23] M. Ji, Y. Sun, J. H. M. Danilevsky, and J. Gao, "Graph regularized transductive classification on heterogeneous information networks," in *Proc. 24th Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2010, pp. 570–586.
[24] B. Chen, D. Ying, and D. J. Wild, "Assessing drug target association using semantic linked data," *Plos Comput. Biol.*, vol. 8, no. 7, 2012, Art. no. e1002574.
[25] W. Gatterbauer, S. Günnemann, D. Koutra, and C. Faloutsos, "Linearized and single-pass belief propagation," *Proc. VLDB Endowment*, vol. 8, no. 5, pp. 581–592, 2015.
[26] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting delays in software projects using networked classification," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2015, pp. 353–364.
[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://dblp.org/rec/bib/conf/iclr/KipfW17
[28] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, 2019, pp. 2022–2032. [Online]. Available: http://doi.acm.org/10.1145/3308558.3313562
[29] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. 32nd AAAI Conf. Artif. Intell., (AAAI-18), 30th Innovative Appl. Artif. Intell. (IAAI-18), 8th {AAAI} Symp. Educ. Adv. Artif. Intell. (EAAI-18)*, New Orleans, Louisiana, USA, Feb. 2–7, 2018, pp. 3538–3545. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16098
[30] V. D. M. Laurens and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 2605, pp. 2579–2605, 2008.
[31] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 593–598.
[32] C. Loglisci, A. Appice, and D. Malerba, "Collective regression for handling autocorrelation of network data in a transductive setting," *J. Intell. Inf. Syst.*, vol. 46, no. 3, pp. 447–472, 2015.
[33] Q. Lu and L. Getoor, "Link-based classification," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 496–503.
[34] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Tech. Rep., Stanford InfoLab, 1999. [Online]. Available: https://www.bibsonomy.org/bibtex/2eb5a6b6671b4dd97e6921da016f85993/albinzehe
[35] T. Haveliwala, S. Kamvar, and G. Jeh, "An analytical comparison of approaches to personalizing pagerank," Tech. Rep., Stanford, 2003.
[36] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 613–622.
[37] M. Ji, J. Han, and M. Danilevsky, "Ranking-based classification of heterogeneous information networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 1298–1306.
[38] X. Kong, B. Cao, and P. S. Yu, "Multi-label classification by mining label and instance correlations from heterogeneous information networks," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 614–622.
[39] Z. Kou and W. W. Cohen, "Stacked graphical models for efficient inference in markov random fields," in *Proc. 7th SIAM Int. Conf. Data Mining*, 2007, pp. 533–538.
[40] S. Nandanwar and M. N. Murty, "Structural neighborhood based classification of nodes in a network," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1085–1094.
[41] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. 33nd Int. Conf. Mach. Learn.*, New York City, NY, USA, Jun. 19–24, 2016, pp. 2014–2023. [Online]. Available: http://proceedings.mlr.press/v48/niepert16.html
[42] M. Edwards and X. Xie, "Graph based convolutional neural network," *CoRR*, vol. abs/1609.08965, 2016. [Online]. Available: http://arxiv.org/abs/1609.08965
[43] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proc. Neur. Inf. Process. 25th Int. Conf.*, Siem Reap, Cambodia, Dec. 13–16, 2018, pp. 362–373, doi: 10.1007/978-3-030-04167-0_33.
[44] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.
[45] J. Moore and J. Neville, "Deep collective inference," in *Proc. Workshop Statistical Relational Learn., 31st Nat. Conf. Artif. Intell.*, 2017, pp. 2364–2372.
[46] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Deep inductive network representation learning," in *Proc. Companion Web Conf.*, 2018, pp. 953–960.

Yun Xiong received the PhD degree in computer and software theory from Fudan University. She is a professor of computer science at Fudan University, Shanghai, China. Her research interests include database and data mining.

Yizhou Zhang is currently working toward the MS degree in computer science at Fudan University, Shanghai, China. His research interests include graph mining and data mining.


Xiangnan Kong received the BS and MS degrees in computer science from Nanjing University, and the PhD degree in computer science from the University of Illinois at Chicago. He is an assistant professor in computer science at the Worcester Polytechnic Institute in the US. His research interests include data mining and machine learning.

Huidi Chen is currently working toward the MS degree in computer technology at Fudan University, Shanghai, China. His research interests include graph mining and data mining.

Yangyong Zhu received the PhD degree in computer and software theory from Fudan University, China. He is a professor of computer science at Fudan University, Shanghai, China. His research interests include database and data mining.