Deep Collective Classification in Heterogeneous Information Networks

Yizhou Zhang[1] Yun Xiong*[1,2] Xiangnan Kong[3] Shanshan Li[4] Jinhong Mi[1] Yangyong Zhu[1,2]
[1]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University, China
[2]Shanghai Institute for Advanced Communication and Data Science, Fudan University, China
[3]Worcester Polytechnic Institute, Worcester, MA, USA
[4]School of Computer, National University of Defense Technology, China
∗ is corresponding author
(yizhouzhang14,yunx*,mij,yyzhu)@fudan.edu.cn,xkong@wpi.edu,shanshanli@nudt.edu.cn

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 involves 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. The proposed method 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.

KEYWORDS

Collective Classification, Graph Convolution, Heterogeneous Information Networks, Graph Mining, Deep Learning

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04..
https://doi.org/10.1145/3178876.3186106

ACM Reference Format:
Yizhou Zhang[1] Yun Xiong*[1,2] Xiangnan Kong[3] Shanshan Li[4] Jinhong Mi[1] Yangyong Zhu[1,2] . 2018. Deep Collective Classification in Heterogeneous Information Networks. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186106

1 INTRODUCTION

Collective classification [30] aims at exploiting the label autocorrelation among a group of inter-connected instances and predicting their class labels collectively. 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, 23, 27, 30].

Previous works on collective classification are concentrated on conventional relational 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 [19] offer automatic end-to-end feature learning in a variety of domains, such as vision [5], speech and NLP [32]. However, current research on deep learning mainly focuses 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 collective classification* in HINs, which involves a hierarchy of different types of autocorrelations among the instances. For example, in Figure 1, we show a deep relational learning task, *i.e.*, predicting the research area (label) of the authors in *bibliographic networks*. Different authors are not only explicitly inter-connected through co-author relations, but also implicitly connected through latent relations, such as adviser-advisee, share-advisor, colleague. The dotted lines in Figure 1 represent implicit relationships between authors that exist in real world, but can only be inferred in the DBLP network.

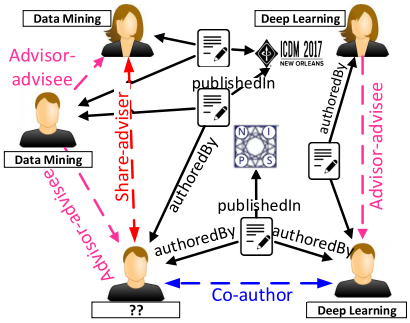


Figure 1: An example of *simple* links and *hidden* links in DBLP network. Here the solid lines represent the *simple* relations exist in DBLP network, and the dotted lines represent the *hidden* relations that exist in real world but should be inferred from DBLP network, *i.e.*, co-author (blue line),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

Although there exist previous methods on collective classification in HINs [16, 37], deep collective classification 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. The complex relationships are not given directly by the links in the network, but can be inferred by a hierarchy of relational features. For instance, in the DBLP network in Figure 1, there are co-author relations (simple relationships) between authors; advisor-advisee relations (hidden relationships), and share-adviser relations (complex relationships). 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 Figure 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 Figure 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.

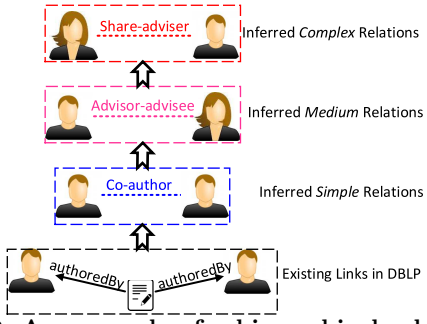


Figure 2: An example of a hierarchical relations between authors in the DBLP network. Here the complex/obscure relations (above) can be inferred from the existing/simple relations (bottom).

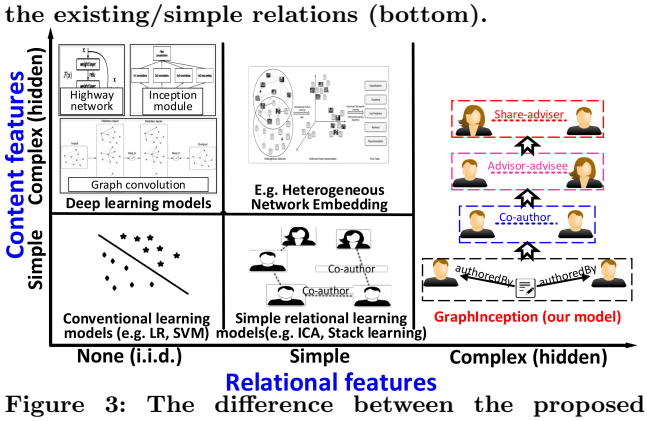


Figure 3: The difference between the proposed method and conventional methods. We divided the algorithms according to the difficulty of the content features and the relational features. In particular, the deep learning models include highway network [33], inception module [35], graph convolution model [6], and HNE [2].

• **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 properties of different types of nodes (links) are very different, which makes it difficult to apply deep learning models directly. For example, graph convolution models [6], assume each node in the network sharing the same convolution kernels, which is untenable in HINs. Although there exist some researches for collective classification in HINs [16, 37], however, most of them are shallow models which ignore the deep relational features in the network.

In order to address the above challenges, we present a deep graph convolutional model, called GraphInception, for collective classification in HINs. Figure 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 [35], a highly efficient deep convolutional model for CNN with fewer parameters and deeper layers.

Table 1: Important Notations.

Symbol	Definition
$G = (\mathcal{V}, \mathcal{E})$	A heterogeneous information network.
$\mathcal{V} = \{\mathcal{V}_1, \cdots, \mathcal{V}_m\}$	The set of nodes, involving m types of nodes. The target node type is $\mathcal{V}_1$.
$\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$	The set of local features for all instances in $\mathcal{V}_1$.
$\mathcal{Y} = \{Y_1, \cdots, Y_n\}$	The set of label values for all instances in $\mathcal{V}_1$, $Y_i \in \{1, \cdots, C\}$.
$\mathcal{L}$ and $\mathcal{U}$	The training set and test set, where $\mathcal{L} \bigcup \mathcal{U} = \mathcal{V}_1$, $\mathcal{L} \bigcap \mathcal{U} = \emptyset$.
$\mathcal{S} = \{\mathcal{P}_1, \cdots, \mathcal{P}_{\lvert\mathcal{S}\rvert}\}$	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 t-th layer in neural networks. T is the number of convolutional layers.

2 PRELIMINARIES

Notation convention: We use upper-case letters for matrices, bold lower-case 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* [34].

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: $\mathcal{V}_1 = \{v_{11}, \cdots, v_{1n_1}\}, \cdots, \mathcal{V}_m = \{v_{m1}, \cdots, v_{mn_m}\}$, where v_{ji} represents the i-th 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 Figure 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 [9], 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 $Y_i \in \{1, \cdots, C\}$ indicating the class label assigned to node v_{1i}. $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ and $\mathcal{Y} = \{Y_1, \cdots, Y_n\}$ 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 [30]. Let $\mathcal{N}_i$ ($\mathcal{N}_i \subseteq \mathcal{V}_1$) represents the set of nodes associated with v_{1i}, $\mathcal{Y}_{\mathcal{N}_i} = \{Y_i | v_{1i} \in \mathcal{N}_i\}$. Then the task of *collective classification in HINs* is to estimate:

$$\Pr(\mathcal{Y}_\mathcal{U} | \mathcal{X}, \mathcal{Y}_\mathcal{L}) \propto \prod_{v_{1i} \in \mathcal{U}} Pr(Y_i | \mathbf{x}_i, \mathcal{Y}_{\mathcal{N}_i}) \quad (1)$$

It is challenging to learn and infer about $Pr(Y_i | \mathbf{x}_i, \mathcal{Y}_{\mathcal{N}_i})$, 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.

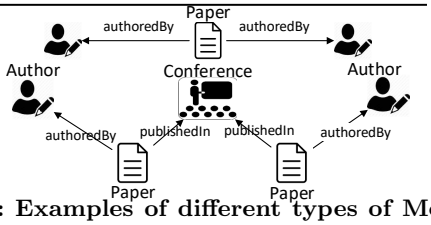


Figure 4: Examples of different types of Meta-path between two authors in DBLP Network.

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 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 Figure 5.

3.1 Multi-channel Network Translation

There are abundant types of nodes in a HIN, while the diversities between different node types 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* [34], 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.

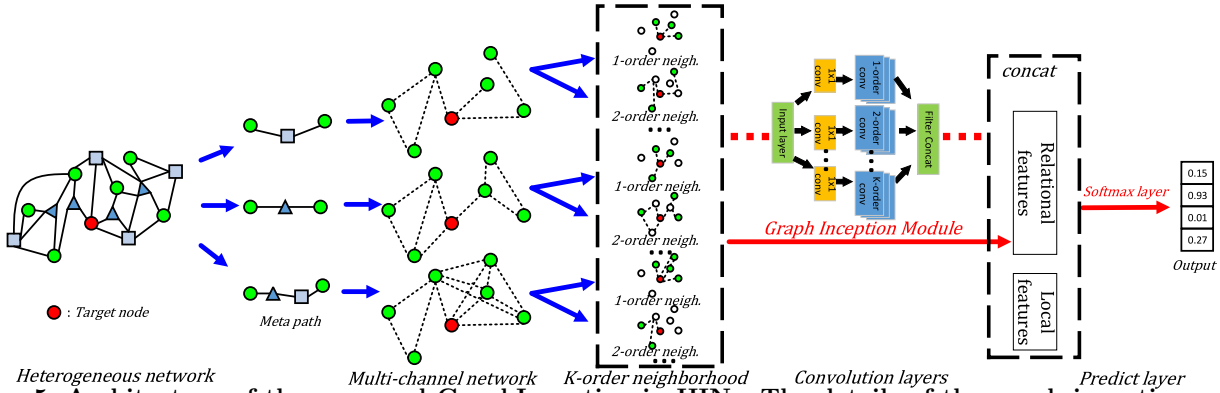


Figure 5: Architecture of the proposed GraphInception in HINs. The details of the graph inception module is shown in Figure 6.

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 Figure 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 Figure 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 $\mathcal{S} = \{\mathcal{P}_1, \cdots, \mathcal{P}_{|\mathcal{S}|}\}$, the translated multi-channel network G' is defined as:

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

where $\mathcal{E}_{1\ell} \subseteq \mathcal{V}_1 \times \mathcal{V}_1$ denotes the links between the instances in $\mathcal{V}_1$, which 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 [16] and greedy tree-based model [24]. 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 [6, 10, 22]. 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 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.*,

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

where $\mathbf{U}$ is the eigenvector matrix of $\mathbf{P}$, $\mathbf{\Lambda}$ is the diagnal matrix of eigenvalues of $\mathbf{P}$, $g_\theta(\mathbf{\Lambda})$ 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 $\mathbf{\Lambda}$ are complex matrices because $\mathbf{P}$ is unsymmetrical matrix. However, recent work [8] 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 [6]:

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

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

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

Note that this expression is now K-order neighborhood since it is a K-th 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

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 $\mathbf{L}$	Transition probability matrix $\mathbf{P}$
Convolution Input	Neighbor nodes with itself	Neighbor nodes only
Convolution Output	Content features	Relational features
Networks	Homogeneous networks	Heterogeneous networks

each channel of G', *i.e.*,

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

where $\mathbf{P}_\ell$ ($\ell = 1, \cdots, |\mathcal{S}|$) 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:

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

$\Theta_{\ell k} \in \mathbb{R}^{C\times F}$ ($\ell = 1, \cdots, |\mathcal{S}|$) is now a matrix of filter parameters. A function r of a vector $\mathbf{x}$ is defined as $r(\mathbf{x}) = (r(x_1), \cdots, r(x_n))$, here $r(x_i) = \max(0, x_i)$ is the Relu function. The i-th 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 [35], 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 Figure 6(a), 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 Figure 6(b), 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 t-th layer is defined as:

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

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 t-th layer of the neural 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, $\Theta_{\ell k}^t \in \mathbb{R}^{C\times F}$ ($k = 1, 2$), otherwise $\Theta_{\ell k}^t \in \mathbb{R}^{F\times F}$. To reduce the number of parameters, we replace $C_{\ell 2}^t$ as:

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

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

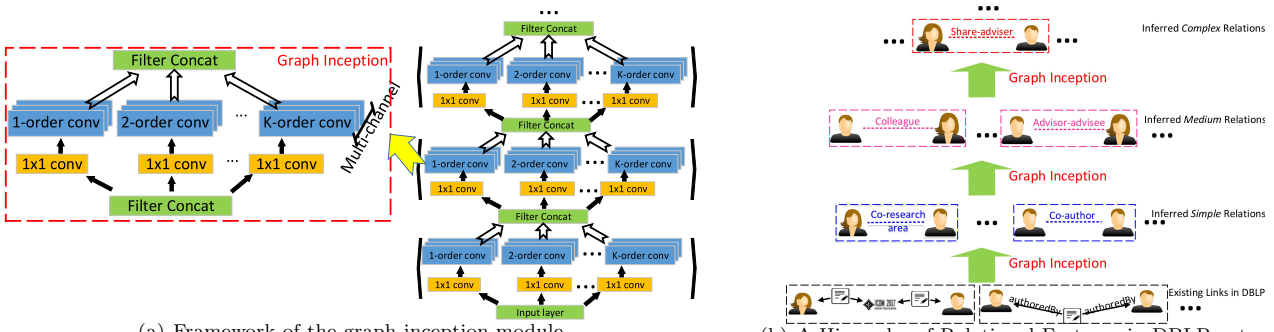
Compared with Eq. 7, the graph inception module-based method has three strengths: 1) It can significantly reduce the required storage space when K is a large number (Eq. 7 need to store K matrices while Eq. 8 only need to store $\mathbf{P}$ and $\mathbf{P}^2$); 2) It requires fewer parameters than Eq. 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 PROPOSED 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.*,

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



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

Figure 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).

Input:
G : a heterogeneous network, T : depth of inception layer
Max_{It}: maximum # of iterations, $\mathcal{X}$: features for all instances
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, $\textbf{length}(\mathcal{P}_\ell) \leqslant p_{max}$.
- 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: $\mathbf{x}'_i = (\mathbf{x}_i, \textbf{vec}(\mathbf{H}_i^T))$, 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:
$\mathcal{Y}_\mathcal{U} = (Y_1, \cdots, Y_{|\mathcal{U}|})$: the labels of test instances ($v_{1i} \in \mathcal{U}$).

Figure 7: The GraphInception algorithm

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 $c \in \{1, \cdots, C\}$ indicates the label of nodes. Here $\textbf{vec}(\cdot)$ 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 label of the target node and the local features of the neighboring nodes.

Inspired by the success of iterative classification method [30], we propose an algorithm, called GraphInception, to solve the collective classification problem in HINs. The framework of GraphInception is shown in Figure 7. The algorithm includes following steps:

Multi-channel Network Construction: Given a HIN G, we first extract a meta-path set $\mathcal{S} = \{\mathcal{P}_1, \cdots, \mathcal{P}_{|\mathcal{S}|}\}$ 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 $\mathbf{x}'_i = (\textbf{vec}(\mathbf{H}_i^T), \mathbf{x}_i)$ 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.

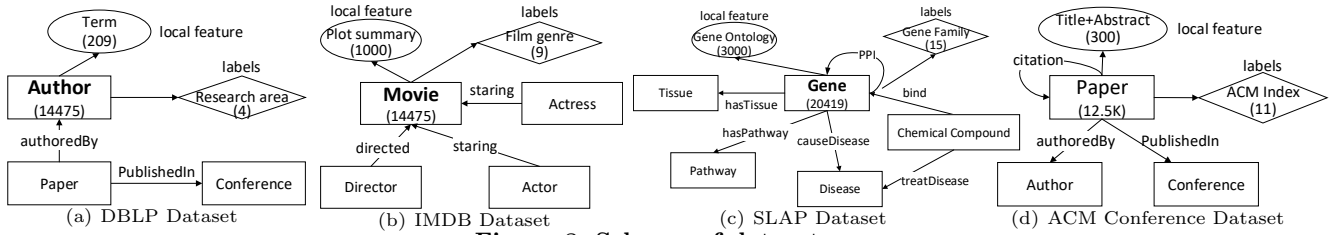
Iterative Inference: In the inference step, we iteratively update the label values of neighboring nodes based on latest predict results, and then use these new labels to make a prediction. The iterative process terminates when the convergence criteria are met. In the end, we will get $\mathcal{Y}_\mathcal{U}$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

Our proposed GraphInception model can not only be applied to ICA framework, but can also be easily extended to other collective classification frameworks including stack learning and label propagation by replacing the traditional relational features with Eq. 8.

6 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, 11, 20, 27, 30] of relational data, has been investigated by many researchers. Basic collective classification problems mainly focus on homogeneous network [21, 23]. Ji [12] studied a specialized classification problem on HINs, where different types of nodes share a same set of label concepts. Kong et al. [15, 16, 37] proposed methods based on meta-path to solve the collective classification problem on one-type nodes in HINs. Kou [17] proposed a method based on stacked model to solve collective classification problem. Choetkiertikul [4] 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 [26] proposed a deep random walk-based collective classification model, but



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

Figure 8: Schema of datasets.

Table 3: Summary of datasets.

Dataset	Node Type	Link Type	Feature	Label	Instance
DBLP	3	2	209	4	14.5K
IMDB	4	3	1000	9	18.4K
SLAP	5	6	2695	15	20.4K
ACM	3	3	300	11	12.5K

it focuses on homogeneous networks. In this work, we chose three representative algorithms in [4, 16, 30] as competing algorithms.

On the other hand, in deep learning models, there exist many related works on graph convolution recently [6, 7, 10, 28, 31]. 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 [6]. 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 [6, 7] tried some localized filters to overcome the problem, which inspired our algorithm. There also exist other deep models which are applied to collective classification problem. Wang [36] proposed a deep network embedding method, which can be used to solve the classification problem on networks. Kipf [14] proposed a semi-supervised classification method based on graph convolution, while Moore [25] proposed a semi-supervised classification method based on RNN model. However, all of them focused on homogeneous networks. Chang [2] proposed a deep architecture for heterogeneous network embedding, which can be used to classified nodes in HINs. Pham [29] proposed a collective classification method on multi-relational networks based on stacked model [17] and highway networks [33]. In this work, we also chose three representative algorithms in [14, 29, 33] as competing algorithms.

7 EXPERIMENTS

7.1 Data Collection

In order to validate the collective classification performances, we apply our algorithm to four real-world HINs. Note that the IMDB dataset is a multi-label dataset, and the rest are multi-class datasets. (Summarized in Table 3).

•**DBLP Dataset:** The first dataset, *i.e.*, DBLP four areas [13], 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 [13], and the network schema is shown in Figure 8(a).

•**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 Figure 8(b). For detailed description of the IMDB dataset, please refer to [29].

•**SLAP Dataset:** The third dataset is a bioinformatic dataset SLAP [3]. As showed in Figure 8(c), 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 [16].

•**ACM Conference Dataset:** The last dataset is also a bibliographic information network, ACM Conference dataset[3], and the network schema is shown in Figure 8(d). 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 [16].

7.2 Compared Algorithms

To demonstrate the effectiveness of our method, we compare with the following state-of-the-art algorithms (summarized in Table 4):

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

Table 5: Results on the DBLP, SLAP and ACM Conference datasets. "↓" indicates the smaller the value the better performance; "↑" indicates the larger the value the better performance.

(a) Results (mean (rank)) on the DBLP dataset.(b) Results (mean (rank)) on the SLAP dataset.(c) Results (mean (rank)) on the ACM dataset.

Algorithms	Evaluation Criteria		Ave.
	Acc.↑	F1↑	Rank
GraphInception	**0.923** (1)	**0.917** (1)	1
Stacked learning	0.916 (2)	0.911 (2)	2
CLN	0.912 (3)	0.906 (3)	3
HCC	0.906 (4)	0.898 (4)	4
ICA	0.807 (5)	0.799 (5)	5
GCN	0.803 (6)	0.797 (6)	6
GCN (metapath)	0.794 (7)	0.788 (7)	7
Highway network	0.781 (8)	0.773 (8)	8
LR	0.776 (9)	0.770 (9)	9

Algorithms	Evaluation Criteria		Ave.
	Acc.↑	F1↑	Rank
GraphInception	**0.349** (1)	**0.317** (1)	1
HCC	0.345 (2)	0.312 (2)	2
ICA	0.344 (3)	0.310 (3)	3
Highway network	0.338 (4)	0.307 (4)	4
LR	0.337 (5)	0.305 (5)	5
Stacked Learning	0.297 (7)	0.216 (6)	6.5
GCN	0.263 (8)	0.222 (6)	7
CLN	0.282 (7)	0.191 (8)	7.5
GCN (metapath)	0.182 (9)	0.175 (9)	9

Algorithms	Evaluation Criteria		Ave.
	Acc.↑	F1↑	Rank
GraphInception	0.733 (2)	**0.369** (1)	1.5
HCC	**0.740** (1)	0.298 (5)	3
GCN (metapath)	0.664 (5)	0.366 (2)	3.5
Stacked learning	0.724 (3)	0.310 (4)	3.5
ICA	0.682 (4)	0.233 (6)	5
GCN	0.618 (8)	0.314 (3)	5.5
LR	0.653 (6)	0.212 (7)	6.5
Highway network	0.624 (7)	0.176 (8)	7.5
CLN	0.521 (9)	0.065 (9)	9

Table 4: Types of models, based on the kinds of features used.

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

• Logistic Regression (LR for short): baseline algorithm.
• Highway Network [33]: A type of neural network 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 [30]: A basic collective classification method in homogeneous networks.
• HCC [16]: This is a collective classification approach, which works on HIN by exploiting dependencies based on multiple meta paths in the network.
• Stacked Learning [4]: 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) [29]: This is a deep feed-forward network for collective classification in multi-relational domains, with shared parameters for each layer. It also implements the highway layer into the model.
• GCN [14]: 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.
• 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 maximum 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 metapath length p_{max} is 4. The hidden dimension of the convolutional 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: $1, 2, 3, 4$. 5-fold cross validation is used in all experiments, and the results are reported by the mean results of 10 runs. Code for our model can be found on Github[4].

Table 6: Results (mean (rank)) on the IMDB dataset. "↓" indicates the smaller the value the better performance; "↑" indicates the larger the value the better performance.

Algorithms	Evaluation Criteria			Ave.
	Hamming loss↓	Micro F1↑	Subset 0/1 loss ↓	Rank
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 (4)	0.550 (2)	0.945 (4)	3.33
ICA	0.317 (5)	0.524 (4)	0.957 (6)	5
LR	0.341 (6)	0.503 (6)	0.967 (7)	6.33
CLN	0.406 (8)	0.417 (9)	0.954 (5)	7.33
GCN	0.388 (7)	0.479 (7)	0.968 (8)	7.33
Highway network	0.481 (9)	0.435 (8)	0.994 (9)	8.67

7.3 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 5. 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 5 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 outperform than Highway network and ICA method. These

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

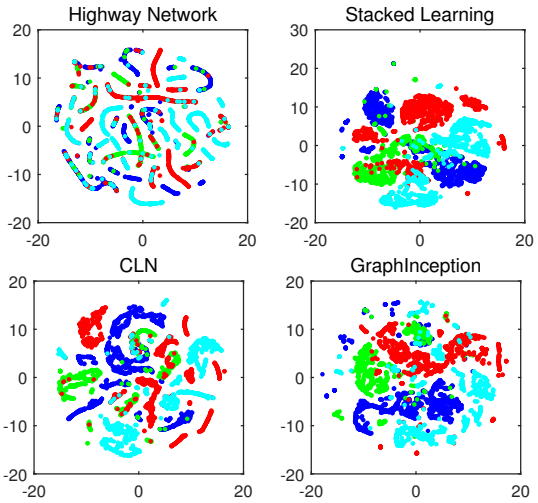


Figure 9: Visualize the hidden layer activations on DBLP dataset.

results support that the heterogeneous dependencies among instances can improve classification performance. 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.

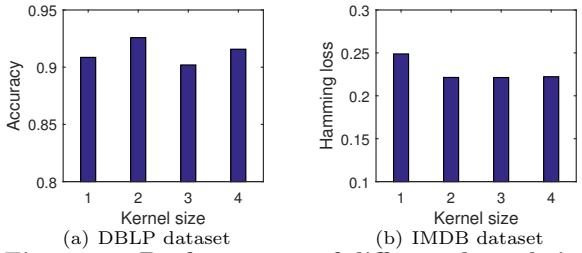
7.4 Multi-label Classification Performance

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 6. 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 6 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 (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.

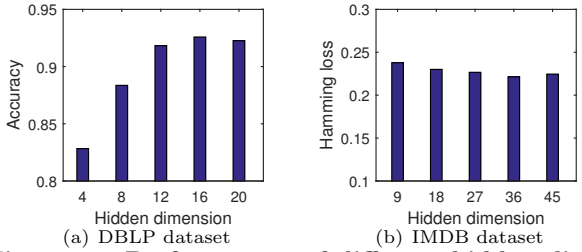
7.5 Relational Features Visualization

To better understand the learned relational features of our model, we use $t-SNE$ [18] to visualize the hidden layer activations of Eq. 8 be trained on DBLP dataset. We also compare the visualization results with other deep learning models, including: highway network, stacked learning and



(a) DBLP dataset (b) IMDB dataset

Figure 10: Performances of different kernel sizes.



(a) DBLP dataset (b) IMDB dataset

Figure 11: Performances of different hidden dimension.

CLN. The results are shown in Figure 9, colors denote research area class of authors. We can find 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 with GCN which utilizes both local features and neighbor nodes labels in hidden layer, while other models (including our model) only use neighbor nodes labels in hidden layer, which is unfair to other models.

7.6 Parameters Sensitivity

There exist two essential hyper-parameters in GraphInception: the convolutional kernel size K and the hidden dimension of convolutional filters F. To test the stability of the performances of GraphInception method, we test different values of K and F 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 Figure 10 and Figure 11. In Figure 10, we can find that it is not sensitive to the kernel sizes on both DBLP dataset and IMDB dataset. In Figure 11, we can find that more hidden filters can achieve better performance on both DBLP dataset and IMDB dataset.

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.

9 ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China Projects No.91546105, No.U1636207, the National Science Foundation through grand IIS-1718310, the National High Technology Research and Development Program of China No.2015AA020105, the Shanghai Science and Technology Development Fund No.16JC1400801, No.17511105502, No.17511101702, No.16511102204, Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase) under Grant No.U1501501.

REFERENCES

[1] I. Alodah and J. Neville. Combining gradient boosting machines with collective inference to predict continuous values. *CoRR*, abs/1607.00110, 2016.
[2] S. Chang, W. Han, J. Tang, G. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *KDD*, 2015.
[3] B. Chen, D. Ying, and D. J. Wild. Assessing drug target association using semantic linked data. *Plos Computational Biology*, 8(7), 2012.
[4] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose. Predicting delays in software projects using networked classification. In *ASE*, 2015.
[5] A. Coates. Deep learning for machine vision. In *BMVC*, 2013.
[6] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.
[7] M. Edwards and X. Xie. Graph based convolutional neural network. *CoRR*, abs/1609.08965, 2016.
[8] N. Guberman. On complex valued convolutional neural networks. *CoRR*, abs/1602.09046, 2016.
[9] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *TiiS*, 5(4), 2016.
[10] M. Henaff, J. Bruna, and Y. Lecun. Deep convolutional networks on graph-structured data. *CoRR*, abs/1506.05163, 2015.
[11] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *KDD*, 2004.
[12] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *KDD*, 2011.
[13] M. Ji, Y. Sun, J. H. M. Danilevsky, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *PKDD*, 2010.
[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
[15] X. Kong, B. Cao, and P. S. Yu. Multi-label classification by mining label and instance correlations from heterogeneous information networks. In *KDD*, 2013.
[16] X. Kong, P. S. Yu, Y. Ding, and D. J. Wild. Meta path-based collective classification in heterogeneous information networks. In *CIKM*, 2012.
[17] Z. Kou and W. W. Cohen. Stacked graphical models for efficient inference in markov random fields. In *SDM*, 2007.
[18] V. D. M. Laurens and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(2605), 2008.
[19] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553), 2015.
[20] C. Loglisci, A. Appice, and D. Malerba. Collective regression for handling autocorrelation of network data in a transductive setting. *J. Intell. Inf. Syst.*, 46(3), 2015.
[21] Q. Lu and L. Getoor. Link-based classification. In *ICML*, 2003.
[22] S. Mallat. *A wavelet tour of signal processing (2. ed.)*. Academic Press, 1999.
[23] L. K. Mcdowell and D. W. Aha. Labels or attributes?: rethinking the neighbors for collective classification in sparsely-labeled networks. In *CIKM*, 2013.
[24] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang. Discovering meta-paths in large heterogeneous information networks. In *WWW*, 2015.
[25] J. Moore and J. Neville. Deep collective inference. In *AAAI*, 2017.
[26] S. Nandanwar and M. N. Murty. Structural neighborhood based classification of nodes in a network. In *KDD*, 2016.
[27] J. Neville and D. Jensen. Iterative classification in relational data. In *AAAI*, 2000.
[28] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. *CoRR*, abs/1605.05273, 2016.
[29] T. Pham, T. Tran, D. Q. Phung, and S. Venkatesh. Column networks for collective classification. In *AAAI*, 2017.
[30] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3), 2008.
[31] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. *CoRR*, abs/1612.07659, 2016.
[32] R. Socher, Y. Bengio, and C. D. Manning. Deep learning for NLP (without magic). In *ACL*, 2012.
[33] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.
[34] Y. Sun and J. Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers, 2012.
[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
[36] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *KDD*, 2016.
[37] Y. Zhang, Y. Xiong, X. Kong, and Y. Zhu. Netcycle:collective evolution inference in heterogeneous information networks. In *KDD*, 2016.