Conditional Random Field Enhanced Graph Convolutional Neural Networks

Hongchang Gao
Electrical & Computer Engineering
University of Pittsburgh
Pittsburgh, USA
JD Finance America Corporation
hongchanggao@gmail.com

Jian Pei
School of Computing Science
Simon Fraser University
Burnaby, Canada
JD.com
jpei@cs.sfu.ca

Heng Huang*
Electrical & Computer Engineering
University of Pittsburgh
Pittsburgh, USA
JD Finance America Corporation
heng.huang@pitt.edu

ABSTRACT

Graph convolutional neural networks have attracted increasing attention in recent years. Unlike the standard convolutional neural network, graph convolutional neural networks perform the convolutional operation on the graph data. Compared with the generic data, the graph data possess the similarity information between different nodes. Thus, it is important to preserve this kind of similarity information in the hidden layers of graph convolutional neural networks. However, existing works fail to do that. On the other hand, it is challenging to enforce the hidden layers to preserve the similarity relationship. To address this issue, we propose a novel CRF layer for graph convolutional neural networks to encourage similar nodes to have similar hidden features. In this way, the similarity information can be preserved explicitly. In addition, the proposed CRF layer is easy to compute and optimize. Therefore, it can be easily inserted into existing graph convolutional neural networks to improve their performance. At last, extensive experimental results have verified the effectiveness of our proposed CRF layer.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**; Neural networks.

KEYWORDS

Graph convolutional neural networks; Conditional random field; Similarity

ACM Reference Format:
Hongchang Gao, Jian Pei, and Heng Huang. 2019. Conditional Random Field Enhanced Graph Convolutional Neural Networks. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330888

*To whom all correspondence should be addressed. This work was partially supported by U.S. NSF IIS 1836945, IIS 1836938, DBI 1836866, IIS 1845666, IIS 1852606, IIS 1838627, IIS 1837956.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330888

1 INTRODUCTION

In recent years, deep convolutional neural networks have achieved great success in a wide variety of tasks, such as image classification [18, 25], image generation [4, 13, 25], and machine translation [36]. The basic idea of convolutional neural networks is to conduct the convolutional operation in a local neighborhood to explore the local correlation. For image data, there is a grid-like structure so that the implicit spatial order makes it easy to perform the convolutional operation in the local neighborhood. In practical applications, graph is a natural representation of numerous real-world data, such as social networks, knowledge graphs, citation networks. However, graphs have no the regular grid-like structure so that it is difficult to determine the local neighborhood to conduct the convolutional operation. To address this issue, the graph convolutional neural network has been proposed, which is designed to perform convolution on the complicated graph data. It has shown superior performance in various tasks, such as node classification [23], recommender system [44]. As a result, graph convolutional neural networks have attracted much attention in recent years.

Generally speaking, graph convolutional neural networks can be categorized into two classes: spatial approaches and spectral approaches. Specifically, spatial approaches [2, 11, 17, 34] conduct convolutions directly on the graph. More specifically, this kind of methods first construct a fixed-size neighborhood for each node in the graph and then perform regular convolution on this neighborhood. For instance, [34] proposes to construct the neighborhood from a fixed-size node sequence and then conduct the convolutional operation. Spectral approaches [5, 10] perform the convolutional operation in the spectral domain. In this way, spectral approaches do not need to construct the neighborhood explicitly for the complicated graph data. For instance, [5] defines the convolution in the Fourier domain which can be done in the eigenspace of the graph Laplacian. [10] proposes the fast localized spectral filtering for graph convolutional neural networks to avoid the expensive eigendecomposition. Recently, [23] further simplifies the spectral approach by directly aggregating the 1-hop neighboring nodes. With these development, graph convolutional neural networks have been successfully applied to a couple of tasks [21, 23, 27, 28, 33, 38, 41, 46], such as node classification [23, 41], relation extraction [46].

Although the aforementioned graph convolutional neural networks can deal with the graph data, yet they fail to fully utilize the properties of graphs. For the graph data, there exist edges which indicate the similarity relationship between different nodes. Two connected nodes imply that they are similar to each other while

disconnected nodes indicate the dissimilarity between them. Existing graph convolutional neural networks can utilize this kind of connectivity information when performing the convolutional operation. For instance, the GCN proposed in [23] aggregates the 1-hop neighboring nodes such that the connectivity information is encoded into the new representation. On the other hand, the connectivity (similarity) information should also be preserved in the new representation (hidden features) obtained from the convolutional operation. However, although the convolutional operation can incorporate the connectivity information, yet it cannot guarantee the obtained hidden features preserve the similarity relationship explicitly. If this kind of relationship is violated in the hidden features, the downstream tasks will be degenerated severely. Thus, it is important and necessary to preserve the similarity information explicitly in the hidden layers of graph convolutional neural networks.

To preserve the similarity relationship in the new representation, numerous methods [1, 19, 37] have been proposed. The most widely used method is the Laplacian regularization, which has been used for various tasks, such as manifold learning [1, 19]. However, this method usually requires the expensive eigendecomposition. Thus, it is not suitable for large-scale neural networks. On the other hand, since we need to enforce the hidden layers of the graph convolutional neural network to satisfy the similarity constraint, it is necessary and important to use a lightweight operation which has small computational overhead and is easy to optimize by the backpropagation. Hence, it is challenging to restrict the behavior of the hidden layers of the graph convolutional neural network.

To address the aforementioned issues, in this paper, we propose a novel CRF layer to regularize the standard graph convolutional neural network to preserve the similarity relationship. Specifically, we resort to the CRF model to restrict the hidden feature of the graph convolutional layer. Then, we find that the solution of the CRF model can be viewed as an individual layer to encourage similar nodes to have similar hidden representations. As shown in Figure 1, this novel CRF layer can be inserted into standard graph convolutional neural networks to regularize the output of the convolutional operation. Furthermore, the proposed CRF layer is easy to compute and optimize. Thus, it is an efficient regularization layer to regularize the behavior of the hidden layers of graph convolutional neural networks. At last, we summarize the contribution of this paper as follows:

- We propose a novel CRF layer for graph convolutional neural networks to encourage similar nodes to have similar hidden features.
- The proposed CRF layer is easy to compute and optimize. It can be inserted into existing graph convolutional neural networks easily.
- Extensive experimental results have verified the effectiveness of our proposed CRF layer.

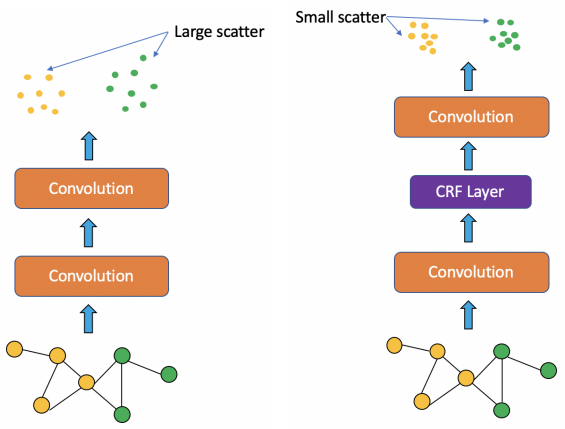
2 RELATED WORKS

In this section, we will review related works about existing graph convolutional neural networks.

Recently, deep learning for the graph data has attracted increasing attention. A wide variety of methods [2, 5, 10–12, 14, 15, 17, 23, 34] have been proposed for this task. Among them, the graph convolutional neural network becomes more and more popular. Essentially, it is to conduct the convolutional operation on the non-Euclidean graph data. Compared with the regular convolutional neural network, there exists a challenge since different nodes in the graph have different size of the neighborhood while the regular convolutional operation requires a fixed-size neighborhood for each node. To address this issue, different approaches [1, 2, 5, 8, 10, 11, 17, 20, 23, 24, 34, 48] have been proposed. In terms of how to perform convolution, existing works can be categorized into two classes: spatial approaches and spectral approaches.

Spatial approaches [2, 11, 17, 34] conduct convolution directly on the graph. Recall that the regular convolution requires that the size of the neighborhood is fixed and nodes in the neighborhood are ordered. However, different nodes in a graph usually have different size of neighborhoods and have no ordering information. Thus, the goal of spatial approaches is to construct a fixed-size and ordered neighborhood to conduct the generic convolutional operation. For instance, [34] first selects a fixed-size sequence of nodes and then constructs the desired neighborhood from this sequence. As a result, the standard convolutional operation can be performed on this fixed-size neighborhood. [2] proposes a parametric construction method to obtain the neighborhood. [17] introduces an inductive method which randomly selects a fixed-size neighborhood and then aggregates the feature of these nodes in a specific manner.

Unlike spatial approaches, spectral approaches [5, 10, 23] try to perform the convolutional operation in the spectral domain rather than the spatial domain. In this way, spectral approaches do not require to explicitly construct the fixed-size neighborhood. For instance, [5] defines the convolution in the Fourier domain which can be done in the eigenspace of the graph Laplacian. However, the



(a) Standard Graph Convolutional Neural Network (b) Graph Convolutional Neural Network with CRF Layer

Figure 1: The comparison between the standard graph convolutional neural network and that with the CRF layer. With the proposed CRF layer, the output features are expected to preserve the similarity better than those of the standard GCN.

computational overhead is large due to the eigendecomposition. To address this issue, [10] proposes the fast localized spectral filtering for graph convolutional neural networks. Specifically, this method resorts to the Chebyshev expansion of the graph Laplacian to avoid the eigendecomposition. Thus, it has the same linear computational complexity as the standard CNN. Later, [23] further simplifies the spectral approach. In detail, [23] proposes to restrict the filters to perform only on the 1-hop neighboring nodes. In this way, it only needs to aggregates the features of these neighboring nodes, which is efficient compared with previous works. Then, graph convolutional neural networks become more and more popular and widely used for a wide variety of tasks. Recently, [41] proposes the graph attention network (GAT) which applies the attention mechanism to graph convolutional neural networks. Specifically, compared with standard graph convolutional neural network which aggregates features of neighboring nodes uniformly, GAT assigns different weights according to the importance of the neighboring node to the reference node.

Although the aforementioned methods can apply the convolutional operation to the graph data, yet they share a common drawback. Unlike the standard data, the graph data possess the connectivity information between different nodes. This connectivity information encodes the similarity relationship. In particular, two connected nodes indicate that they are similar, while disconnected nodes imply the dissimilarity between them. Although existing graph convolutional neural networks can utilize this kind of information when conducting the convolutional operation, yet they cannot guarantee the obtained new features from the convolutional operation satisfy the similarity constraint. Thus, if the new features violate the implicit constraint, the learned features will degenerate downstream tasks. All in all, it is important and necessary to enforce the learned new features to preserve the similarity relationship.

In fact, to deal with the similarity relationship, various works [1, 19, 37] have been proposed. For instance, the graph regularization method [1, 19] is a widely used approach to restrict the new features to preserve the similarity. This method has been successfully used for conventional machine learning models, such as manifold learning [19]. However, this method requires the expensive eigendecomposition of the graph Laplacian. Thus, it is not efficient for large-scale graphs. In addition, to handle the similarity relationship, the conditional random field (CRF) method is another potential choice. CRF is a probabilistic graphical model which can model the pairwise relationship. It is first proposed in [26] for predicting labels of the sequential data. After that, it has been applied to different tasks to encourage similar data points to have similar predictions. For instance, in the image segmentation task [9], CRF is used to refine the coarse pixel-level prediction by exploring the relationship between each pixel and their contexts. Recently, CRF is also applied to the information retrieval task [7] by modeling the pairwise similarity between the query data point and the gallery data point. Besides, a contemporary work, Conditional Graph Neural Field (CGNF) [30], also applies CRF to the graph convolutional neural network. In particular, CGNF utilizes CRF to exploit the correlation of node labels like the conventional CRF model [9, 26]. Conversely, our method does not use node labels in the CRF part. Thus, our method is totally different from CGNF.

3 PRELIMINARY KNOWLEDGE

In this section, we will provide some preliminary knowledge about the conditional random field which is important for our proposed method.

Conditional random field (CRF) is a probabilistic graphical model, which is first proposed in [26] for predicting labels of the sequential data. Later, CRF is introduced to different tasks for structured prediction, such as image segmentation [9], information retrieval [7]. Essentially, CRF is capable of modeling the pairwise relationship between the reference data point and its context to refine the final prediction. Formally, given the input data x_i, CRF aims at predicting y_i by maximizing the conditional probability as follows:

$$P(y_i|x_i) = \frac{1}{Z(x_i)} \exp(-E(y_i|x_i)) \,, \tag{1}$$

where $Z(x_i)$ denotes the partition function which serves as the normalization factor, $E(y_i|x_i)$ represents the energy function. Here, the definition of y_i depends on the specific task. For instance, in the image segmentation task, y_i represents the label for each pixel. In our task, y_i denotes the new representation.

As for the energy function, it includes two components: the unary energy component and the pairwise energy component. The unary energy function gives the prediction for each individual data point. The pairwise energy function aims at capturing the correlation between the individual data point and its context to regularize the unary energy function. As a result, the prediction for each individual data point can benefit from its own information and its neighboring data points' information. Formally, the energy function is defined as follows:

$$E(y_i|x_i) = \psi_u(y_i, x_i) + \sum_j \psi_p(y_i, y_j, x_i, x_j) \,, \tag{2}$$

where $\psi_u(y_i, x_i)$ denotes the unary energy function while $\psi_p(y_i, y_j, x_i, x_j)$ represents the pairwise energy function. For instance, in the image segmentation task, the unary energy function predicts the label for individual pixels in terms of the property of each pixel. The pairwise energy function provides the context information to encourage similar pixels to have similar label assignment. Finally, the mean-field approximation method is usually used to optimize the CRF model.

Inspired by its capability of capturing the pairwise relationship between the reference data point and its context, we will propose to apply CRF to the graph convolutional neural network to preserve the similarity in hidden layers.

4 METHODOLOGY

Assume the input graph $G = \{A, X\}$ includes the adjacency matrix $A \in \mathfrak{R}^{n \times n}$ and the node feature matrix $X \in \mathfrak{R}^{n \times d}$ where n denotes the number of nodes and d represents the dimension of node features. Specifically, $A = [a_{ij}] \in \mathfrak{R}^{n \times n}$ gives the connectivity information between different nodes where $a_{ij} > 0$ represents there exists an edge between node i and node j, otherwise $a_{ij} = 0$. Based on these terminologies, the standard graph convolutional neural network (GCN) proposed in [23] can be represented as follows:

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l+1)}) \,, \tag{3}$$

where $H^{(l)}$ denotes the representation of the graph nodes in the l-th layer, $W^{(l)}$ stands for the model parameter in the l-th layer, $\sigma(\cdot)$ represents the non-linear activation function. Additionally, $\hat{A}$ represents the normalized graph adjacency matrix $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Here, $\tilde{A} = A + I$ where I is an identity matrix, and $\tilde{D}$ is a diagonal matrix with diagonal elements being $\tilde{D}_{ii} = \sum_{j=1}^{n} \tilde{A}_{ij}$.

Intuitively, GCN in Eq. (3) first aggregates features of neighboring nodes in terms of the adjacency matrix by using $\hat{A}H^{(l)}$ and then conducts the rest transformation. For the graph data, nodes are connected by edges. These edges carry the similarity relationship between different nodes. Thus, to learn effective representations $H^{(l+1)}$ from GCN, it is necessary to enforce $H^{(l+1)}$ to preserve the similarity relationship. Otherwise, $H^{(l+1)}$ cannot fully represent the input graph data, degenerating the downstream task. To address this issue, we will propose a novel method to restrict the behavior of $H^{(l+1)}$.

4.1 CRF Layer

In this paper, following [23, 41], we consider the node classification problem whose objective function is defined as follows:

$$J(W; X, \hat{A}, Y) = \mathcal{L}(Y; F(X, \hat{A}, W)) , \tag{4}$$

where Y denotes the label of nodes, W represents the all model parameters of graph convolutional neural networks, $F(\cdot,\cdot,\cdot)$ indicates the graph convolutional neural network mapping, and $\mathcal{L}(\cdot,\cdot)$ stands for the loss function. Following [6], the objective function of graph convolutional neural networks can be reformulated as the following one with the quadratic penalty:

$$\begin{aligned} J(W; X, \hat{A}, Y) = {} & \mathcal{L}(Y; F_L(H^{(L-1)}, \hat{A}, W^{(L)})) \\ & + \sum_{l=1}^{L-1} \frac{\gamma}{2} \|H^{(l)} - F_l(H^{(l-1)}, \hat{A}, W^{(l)})\|_F^2 , \end{aligned} \tag{5}$$

where $\gamma > 0$, L represents the total number of layers, and $F_l(\cdot,\cdot,\cdot)$ denotes the mapping in the l-the layer. For the GCN defined in Eq. (3), $F_l(\cdot,\cdot,\cdot)$ is $\sigma(\hat{A}H^{(l-1)}W^{(l)})$. In terms of [45], under mild conditions, the solution of Eq. (5) converges to that of Eq. (4) when $\gamma \to \infty$. From this new formulation, it is easy to find that there is no any constraint for $H^{(l)}$ to enforce it to satisfy the similarity constraint that a graph possesses. To address this problem, the high-level idea of our method is to enforce a regularization for $H^{(l)}$ to make it satisfy the similarity relationship. Specifically, our general objective function is defined as follows:

$$\begin{aligned} J(W; X, \hat{A}, Y) = {} & \mathcal{L}(Y; F_L(H^{(L-1)}, \hat{A}, W^{(L)})) \\ & + \sum_{l=1}^{L-1} \frac{\gamma}{2} \|H^{(l)} - F_l(H^{(l-1)}, \hat{A}, W^{(l)})\|_F^2 + \mathcal{R}(H^{(l)}) , \end{aligned} \tag{6}$$

where $\mathcal{R}(\cdot)$ denotes a regularization function for implementing the similarity constraint. In fact, there are numerous works for such kind of regularization in the traditional machine learning and data mining models. However, since our purpose is to restrict the behavior of the layers of graph convolutional neural networks, there exist several challenges:

- The regularization term should be easy to compute. It cannot have expensive computation.
- The regularization term should be friendly to the backpropagation strategy. Then, the neural network can still be optimized in the backpropagation way.
- The regularization term should be easy to plug into existing graph convolutional neural networks.

All in all, it is critical and challenging to get an efficient and effective regularization term.

To address the aforementioned challenges, we resort to the conditional random field (CRF) which can capture the pairwise relationship between different nodes. Specifically, the node representation $H^{(l)}$ is viewed as random variables $\{H_{i\cdot}^{(l)}\}$ where $H_{i\cdot}^{(l)}$ represents the i-th row of $H^{(l)}$, corresponding to the representation of node i. These random variables are conditioned on $\{B_{i\cdot}^{(l)}\}$ where $B_{i\cdot}^{(l)} = F_l(H_{i\cdot}^{(l-1)}, \hat{A}, W^{(l)})$ stands for the preliminary representation of node i obtained from the convolutional operation. Based on these terminologies, we have the following CRF model:

$$P(H^{(l)}|B^{(l)}) = \frac{1}{Z(B^{(l)})} \exp(-E(H^{(l)}|B^{(l)})) , \tag{7}$$

where $Z(\cdot)$ serves as the normalization factor and $E(\cdot)$ is the energy function. As we discussed earlier, the energy function includes two components: the unary energy component and the pairwise energy component. In this paper, the unary function is defined as follows:

$$\psi_u(H_{i\cdot}^{(l)}, B_{i\cdot}^{(l)}) = \|H_{i\cdot}^{(l)} - B_{i\cdot}^{(l)}\|_2^2 . \tag{8}$$

In fact, minimizing this unary function will enforce the node representation $H_{i\cdot}^{(l)}$ to be close to that obtained from the convolutional operation. To capture the similarity relationship between different nodes, the pairwise function is defined as follows:

$$\psi_p(H_{i\cdot}^{(l)}, H_{j\cdot}^{(l)}, B_{i\cdot}^{(l)}, B_{j\cdot}^{(l)}) = g_{ij}\|H_{i\cdot}^{(l)} - H_{j\cdot}^{(l)}\|_2^2 , \tag{9}$$

where g_{ij} denotes the similarity between node i and node j. Intuitively, when g_{ij} is large, minimizing $\psi_p(H_{i\cdot}^{(l)}, H_{j\cdot}^{(l)}, B_{i\cdot}^{(l)}, B_{j\cdot}^{(l)})$ will enforce $H_{i\cdot}^{(l)}$ to be close to $H_{j\cdot}^{(l)}$. Otherwise, it will push $H_{i\cdot}^{(l)}$ away from $H_{j\cdot}^{(l)}$. As a result, similar nodes will be encouraged to have similar representations.

Finally, the energy function for node i is defined as follows:

$$E(H_{i\cdot}^{(l)}|B_{i\cdot}^{(l)}) = \alpha\|H_{i\cdot}^{(l)} - B_{i\cdot}^{(l)}\|_2^2 + \beta \sum_{j \in \mathcal{N}_i} g_{ij}\|H_{i\cdot}^{(l)} - H_{j\cdot}^{(l)}\|_2^2 , \tag{10}$$

where $\mathcal{N}_i$ denotes the neighborhood of node i. Here, we introduce two parameters $\alpha > 0$ and $\beta > 0$ to adjust the importance of the two energy functions. Obviously, the energy function defined in Eq. (10) will enforce the new representation in the l-th layer to be close to that obtained from the convolutional operation and encourage similar nodes to have similar new representations. In addition, comparing with Eq. (6), the pairwise energy function in Eq. (10) actually acts as the regularization to encourage the new representations $H^{(l)}$ to preserve the similarity relationship.

After obtaining the energy function, we need to derive a tractable and efficient updating rule for the new representations $H^{(l)}$. Here, we resort to the mean-field approximation method. The basic idea is to find a simple distribution to approximate $P(H^{(l)}|B^{(l)})$ rather than computing $P(H^{(l)}|B^{(l)})$ exactly. Specifically, we employ the simple

distribution $Q(H^{(l)})$ to approximate $P(H^{(l)}|B^{(l)})$. This simple distribution can be represented by the product of independent marginal distributions as $Q(H^{(l)}) = \prod_{i=1}^{n} Q_i(H_{i\cdot}^{(l)})$. Then, we minimize the KL divergence between the original distribution $P(H^{(l)}|B^{(l)})$ and the simple distribution $Q(H^{(l)})$ as follows:

$$\min \mathrm{KL}(Q(H^{(l)})||P(H^{(l)}|B^{(l)})) \,. \tag{11}$$

As a result, we can get the optimal distribution $Q_i^*(H_{i\cdot}^{(l)})$ as follows:

$$\ln Q_i^*(H_{i\cdot}^{(l)}) = \mathbb{E}_{j\neq i}[\ln P(H_{j\cdot}^{(l)}|B_{j\cdot}^{(l)})] + const \,. \tag{12}$$

According to Eq. (7) and Eq. (10), we can get

$$Q_i^*(H_{i\cdot}^{(l)}) \sim \exp\left(\alpha\|H_{i\cdot}^{(l)} - B_{i\cdot}^{(l)}\|_2^2 + \beta \sum_{j\in\mathcal{N}_i} g_{ij}\|H_{i\cdot}^{(l)} - H_{j\cdot}^{(l)}\|_2^2\right) , \tag{13}$$

which indicates that $Q_i^*(H_{i\cdot}^{(l)})$ is a Gaussian function. As a result, the maximum probability is achieved at the expectation of $Q_i^*(H_{i\cdot}^{(l)})$. Then, by computing its expectation, we have the updating rule for the new representations $H^{(l)}$ as follows:

$$(H_{i\cdot}^{(l)})^{k+1} = \frac{\alpha B_{i\cdot}^{(l)} + \beta \sum_{j\in\mathcal{N}_i} g_{ij}(H_{i\cdot}^{(l)})^k}{\alpha + \beta \sum_{j\in\mathcal{N}_i} g_{ij}} \,. \tag{14}$$

Note that it is an iterative updating rule and $(H_{i\cdot}^{(l)})^k$ denotes $H_{i\cdot}^{(l)}$ in the k-th iteration. After K iterations, we set $H_{i\cdot}^{(l)} = (H_{i\cdot}^{(l)})^K$, which is the final node representations in the l-th layer. From Eq. (14), it can be seen that $H_{i\cdot}^{(l)}$ depends on not only the representation $B_{i\cdot}^{(l)}$ which is obtained from the convolutional operation but also the representation of its neighboring nodes. Especially, when the coefficient g_{ij} is large, which means that node j is more similar with node i, it will contribute more to $H_{i\cdot}^{(l)}$. In this way, similar nodes will have similar representations.

In Eq. (14), we need to compute the coefficient g_{ij} between two nodes. In fact, there are a couple of choices to implement it. In this paper, we employ two approaches, which is shown as follows.

Gaussian The most straightforward approach is to use the Gaussian function to compute g_{ij} as follows:

$$g_{ij} = \exp\left(\frac{B_{j\cdot}^{(l)} B_{i\cdot}^{(l)T}}{\|B_{j\cdot}^{(l)}\|_2 \|B_{i\cdot}^{(l)}\|_2} / \sigma^2\right) , \tag{15}$$

where σ is set as a learnable parameter in this paper. Note that $B_{i\cdot}^{(l)}$ is a *row vector*. Intuitively, a large inner product implies a large similarity. Then, node j will contribute more to the representation of node i. After obtaining g_{ij}, it is easy to update $H_{i\cdot}^{(l)}$ without expensive operations.

Neural Network Another choice to compute g_{ij} is the flexible neural network, which has a larger capability than Gaussian function to deal with the similarity between node i and node j. Specifically, we employ a one-layer MLP to compute it as follows:

$$g_{ij} = \frac{\exp(s_{ij})}{\sum_{k\in\mathcal{N}_i} \exp(s_{ik})} \,, \tag{16}$$

where

$$s_{ij} = (W_\theta B_{i\cdot}^{(l)T})^T (W_\beta B_{j\cdot}^{(l)T}) \,. \tag{17}$$

Here, W_θ and W_β are the weights of neural networks. All these weights are learnable parameters which can be optimized with those of standard graph convolutional neural networks.

Finally, Eq. (14) can serve as an individual layer to plug into the existing graph convolutional neural networks. Specifically, as shown in Figure 1, for the GCN proposed in [23], we may have the following layer configuration:

$$\begin{aligned} &\textit{Conv Layer} : B^{(l)} = \sigma(\hat{A}H^{(l-1)}W^{(l)}) \,, \\ &\textit{CRF Layer} : (H_{i\cdot}^{(l)})^{k+1} = \frac{\alpha B_{i\cdot}^{(l)} + \beta \sum_{j\in\mathcal{N}_i} g_{ij}(H_{i\cdot}^{(l)})^k}{\alpha + \beta \sum_{j\in\mathcal{N}_i} g_{ij}} \,. \end{aligned} \tag{18}$$

Here, we call Eq. (14) as the CRF layer. In addition, for the CRF layer, we initialize $(H_{i\cdot}^{(l)})^0 = B_{i\cdot}^{(l)}$ and set the number of iterations to 2 in our paper.

Obviously, with our proposed CRF layer, similar nodes will be encouraged to have similar representations. Consequently, the similarity relationship can be preserved in the layers of graph convolutional neural networks. In addition, it can also be find that it is friendly to back propagation. Thus, we can still use the backpropagation strategy to optimize this kind of neural networks. At last, we summarize our method in Algorithm 1.

Algorithm 1 Graph convolutional neural network with CRF layer.

Require: the number of iterations K
Ensure: $(H^{(l)})^K$
1: Graph Convolutional Layer

$$B^{(l)} = \sigma(\hat{A}H^{(l-1)}W^{(l)})$$

2: CRF Layer
3: **for** $k = 0, 1, \cdots, K-1$ **do**
4: $\quad (H_{i\cdot}^{(l)})^{k+1} = \frac{\alpha B_{i\cdot}^{(l)} + \beta \sum_{j\in\mathcal{N}_i} g_{ij}(H_{i\cdot}^{(l)})^k}{\alpha + \beta \sum_{j\in\mathcal{N}_i} g_{ij}}$
5: **end for**

4.2 Discussion

Recently, [41] applies the attention mechanism to the graph convolutional neural network. That is the graph attention network (GAT) whose convolutional layer is defined as follows:

$$H_{i\cdot}^{(l+1)} = \sigma(\sum_{j\in\mathcal{N}_i} \alpha_{ij} H_{j\cdot}^{(l)} W^{(l+1)}) \,, \tag{19}$$

where $H_{i\cdot}^{(l+1)}$ denotes the feature of the i-th node, α_{ij} represents the attention coefficient between the i-th node and the j-th node. In this way, if α_{ij} is large, which indicates that the j-th node is an important neighbor to the i-th node, it will contribute more to the new feature of the i-th node.

In our proposed CRF layer, there is also a coefficient g_{ij} which represents the coefficient between two nodes. However, the mechanism of our method is different from that of GAT. Specifically, although both GAT and our method aggregate neighboring nodes in terms of the similarity between different nodes, yet GAT acts before the non-linear activation function while our method happens after the non-linear activation function. Since the non-linear

Table 1: Descriptions of benchmark datasets.

Dataset	# Nodes	#Edges	#Features	#Classes
Citeseer	3,327	4,732	3,703	6
Cora	2,708	5,429	1,433	7
Pubmed	19,717	44,338	500	3

function cannot guarantee to preserve the structure of node distributions, the similarity may not be preserved after the convolutional operation. On the contrary, our method acts on the output of the non-linear activation function directly. Thus, it can guarantee the output of the convolutional layer to preserve the similarity relationship. In addition, our method is an iterative approach which can be dynamically affected by the updating of the neighboring nodes. Thus, our method is more flexible.

5 EXPERIMENTS

In this section, we will conduct extensive experiments to verify the performance of proposed CRF layer.

5.1 Datasets

The datasets used in our experiments are same as [23, 41]. Specifically, there are three datasets: Cora, Citeseer, and Pubmed [39]. The nodes of these networks are documents from different topics. The citation between documents serves as the edges. The content of documents corresponds to node features. Here, node features are represented by the bag-of-words of the corresponding document. The details about these documents are described as follows:

- Citeseer is a research paper citation network. There are 3,327 papers from 6 topics. The total number of citations is 4,732. The dimension of node features is 3,703.
- Cora is also a citation network. It has 2,708 documents from 7 topics. Each document has 1,433 features. The number of citations between different documents is 5,429.
- Pubmed is another citation network. It consists of 19,717 documents from 3 classes. Each document has 500 features.

The statistics of these datasets are summarized in Table 1.

In our experiments, we conduct two tasks to evaluate the performance of our proposed method. They are semi-supervised node classification and supervised node classification. As for the semi-supervised task, following [23, 41], 20 nodes per class are labeled as the training set. 500 nodes are selected as the validation set to conduct model selection. After that, the trained model is evaluated on the testing set which has 1,000 nodes. As for the supervised task, following [8], we keep the same validation and testing sets as the semi-supervised task. The rest nodes are all labeled as the training set.

5.2 Experiment Settings

To evaluate the performance of our proposed method, we compare it with various state-of-the-art methods. Following [41], the following semi-superivsed methods are employed as the baseline methods. They are Label Propagation (LP) [47], Semi-supervised Embedding (SemiEmb) [42], Manifold Regularization (ManiReg) [3], DeepWalk [35], Iterative Classification Algorithm (ICA) [29], Planetoid [43], the graph convolutional neural network with Chebyshev filters [10], and the MoNet method [32].

To show the performance of the CRF layer, we apply it to two existing graph convolutional neural networks, including GCN [23] and GAT [41]. Here, we call them CRF-GCN and CRF-GAT. As for CRF-GCN, we employ the same network configuration as GCN. Specifically, there are two convolutional layers. The dimension of the hidden layer is 16. In our experiments, we insert the CRF layer after the first convolutional layer. All the weights are initialized with the Glorot method [16]. Additionally, we apply dropout [40] to all convolutional layers and the dropout ratio is set to 0.5. Furthermore, the weight decay whose parameter is 0.0005 is employed to regularize model parameters. The Adam [22] optimizer with a learning rate of 0.01 is utilized to optimize CRF-GCN.

As for CRF-GAT, there are also two convolutional layers. Similarly, we insert the CRF layer after the first convolutional layer. The other configuration is almost same with that of the original GAT. Specifically, for Cora and Citeseer, the first convolutional layer has 8 attention heads, each of whom has 8 features. The second convolutional layer has only one attention head and the number of its features equals the number of classes. Similar with CRF-GCN, the weight decay with $\lambda = 0.0005$ is employed to regularize the weights of the neural network. Additionally, the dropout with drop rate being 0.6 is applied to all convolutional layers and the normalized attention coefficients. For Pubmed, there are 8 attention heads at the second convolutional layer. In addition, the weight decay is increased to $\lambda = 0.001$. At last, all models are optimized by Adam [22] optimizer.

5.3 Results and Analysis

5.3.1 Semi-supervised Task. In Table 2, we report the classification accuracy of the semi-supervised task. Here, the results of state-of-the-art methods are extracted from the original GAT [41]. In addition, CRF-GCN-Gaussian denotes that g_{ij} is computed by using the Gaussian function while CRF-GCN-NN denotes that it is computed by using the neural network. So does CRF-GAT. Note that since the dimension of the hidden layer of GAT is 64, we also report the result of GCN-64 which has 64 hidden features either.

From Table 2, it can be seen that the proposed CRF regularization layer does be helpful to improve the performance of existing graph convolutional neural networks. Specifically, both CRF-GCN-Gaussian and CRF-GCN-NN can beat the counterpart GCN for almost all datasets. For instance, CRF-GCN-NN can improve upon the standard GCN by 1.0% and 1.8% for Cora and Citeseer, respectively, which indicates that preserving the similarity relationship is beneficial. As for CRF-GAT-Gaussian, it improves upon the standard GAT by 1.6% and 0.6% for Cora and Citeseer, respectively. This further verifies the effectiveness of our proposed method.

To further show the effectiveness of our proposed method, we visualize the learned features in the last layer of GCN for Citeseer dataset. Specifically, in Figure 2. we plot the learned features of the testing set by using T-SNE [31]. It can be seen that the learned features from our proposed methods have a more compact structure than those learned from the standard GCN. In other words, the similarity relationship does be preserved better than the standard GCN. Thus, our method has a better classification performance.

Table 2: The accuracy of semi-supervised node classification. Our results are marked in bold.

Methods	Cora	Citeseer	Pubmed
ManiReg [3]	0.595	0.601	0.707
SemiEmb [42]	0.590	0.596	0.717
LP [47]	0.680	0.453	0.630
DeepWalk [35]	0.672	0.432	0.653
ICA [29]	0.751	0.691	0.739
Planetoid [43]	0.757	0.691	0.739
Chebyshev [10]	0.812	0.698	0.744
MoNet [32]	0.817	-	0.788
GCN [23]	0.815	0.703	0.790
CRF-GCN-Gaussian (ours)	**0.828**	**0.718**	**0.790**
CRF-GCN-NN (ours)	**0.825**	**0.721**	**0.792**
GCN-64	0.814	0.709	0.790
GAT [41]	0.830	0.725	0.790
CRF-GAT-Gaussian (ours)	**0.846**	**0.731**	**0.791**
CRF-GAT-NN (ours)	**0.841**	**0.726**	**0.790**

(a) GCN (b) CRF-GCN-Gaussian (c) CRF-GCN-NN

Figure 2: The visualization of features from the last layer for Citeseer.

(a) GCN (b) Input of the CRF layer (c) Output of the CRF layer

Figure 3: The visualization of features from the hidden layer of GCN and CRF-GCN-Gaussian for Citeseer.

Furthermore, to show the effect of the CRF layer, we visualize the hidden features of the CRF layer. Specifically, in Figure 3(a), we show the output feature of the first convolutional layer of the standard GCN for Citeseer dataset. In Figure 3(b) and Figure 3(c), we visualize the input and output feature of the CRF layer which follows the first convolutional layer of our proposed CRF-GCN-Gaussian, respectively. Compared Figure 3(b) with Figure 3(c), it can be seen that the output feature of the CRF layer have a more

compact structure than the input. In other words, the CRF layer makes the features more compact to preserve the similarity. All these results have verified the effectiveness of our proposed method.

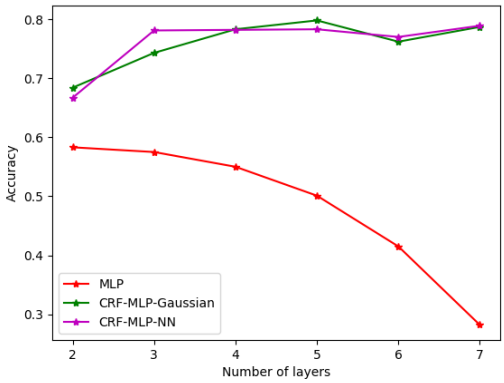


Figure 4: The semi-supervised classification accuracy of different MLP methods with different layers for Cora.

At last, to comprehensively verify the effectiveness of the CRF layer, we apply it to the standard fully connected neural networks. In detail, we insert the CRF layer after each fully connected layer other than the last layer. Here, we call them CRF-MLP-Gaussian and CRF-MLP-NN. In detail, the dimension of the hidden layer is set to 64. The training configuration, such as learning rate, weight decay parameter, dropout ratio, is same with CRF-GCN. In Figure 4, we demonstrate the testing accuracy of the semi-supervised classification for Cora. Here, we show the result of MLP with different layers. It can be seen that CRF-MLP-Gaussian and CRF-MLP-NN can outperform the standard MLP consistently and significantly. IN particular, the standard MLP performs worse when increasing the number of layers while our methods do not. The possible reason is that the hidden layer of the standard MLP cannot preserve the similarity. As a result, the learned representations violate the similarity relationship more severely when stacking more layers. On the contrary, due to the regularization of the CRF layer, our methods perform well when stacking multiple layers. Thus, we can conclude that the proposed CRF layer can preserve the similarity relationship between different nodes effectively.

5.3.2 Supervised Task. To further demonstrate the performance of our proposed method, we conduct the supervised node classification task as [8]. Specifically, we also insert the CRF layer after the first convolutional layer of GCN. Additionally, in this task, we employ the residual connection in the CRF layer as follows:

$$\hat{H}^{(l)} = (H^{(l)})^K + B^{(l)}, \tag{20}$$

where $(H^{(l)})^K$ is the last iteration of the CRF layer.

Similar with the semi-supervised task, we use the early stop strategy to terminate the training procedure when the loss function is not further decreased. The classification result is reported in Figure 5. It can be seen that our proposed methods outperform the standard GCN for almost all cases, which demonstrates the effectiveness of our proposed method.

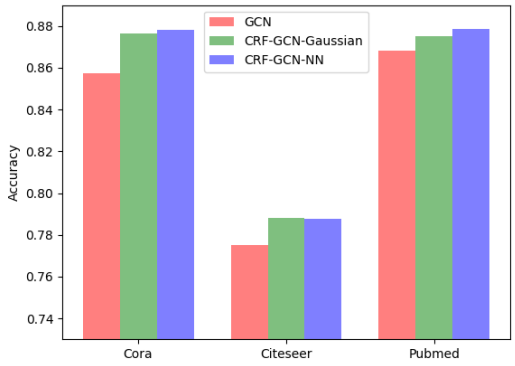


Figure 5: The supervised classification accuracy of different supervised GCN methods.

Table 3: The supervised classification accuracy of MLP.

Dataset	Cora	Citeseer	Pubmed
MLP	0.733	0.757	0.863
CRF-MLP-Gaussian	0.873	0.799	0.886
CRF-MLP-NN	0.852	0.783	0.884

Furthermore, we also apply the proposed CRF layer to fully connected neural networks for the supervised task. Following the configuration of the semi-supervised task, the dimension of hidden layers is also set to 64. In this experiment, there is only one hidden layer and the proposed CRF layer is inserted after the hidden layer. The classification result is shown in Table 3. It can be seen that MLP with the proposed CRF layer significantly outperforms the standard MLP. The underlying reason is that the proposed CRF layer can preserve the similarity relationship in the hidden layer. As a result, the learned representation for nodes is discriminative, benefiting the classification task.

5.3.3 Ablation Study. In the proposed CRF layer, there are two important parameters: α and β, which are used to adjust the importance of $B^{(l)}$ and $H^{(l)}$. In this paper, we set them as the learnable parameters. To show the importance of these parameters, we compare it with the variants: CRF-GCN-Gaussian-const and CRF-GCN-NN-const, which set $\alpha = 1$ and $\beta = 1$. In Table 4, we report the semi-supervised classification accuracy. It can be seen that these two variants degrade the performance significantly for almost all cases. Thus, it is necessary to learn these two parameters automatically.

6 CONCLUSION

In this paper, we propose a novel CRF layer for the graph convolutional neural network. Specifically, by resorting to the CRF model for the hidden layers of graph convolutional neural networks to explore the similarity relationship, we obtain an efficient CRF layer, which can encourage the hidden features to preserve the similarity between different nodes. In addition, the proposed CRF layer is easy

Table 4: The semi-supervised classification accuracy of Cora dataset. The "const" variant represents $\alpha = 1$ and $\beta = 1$.

Dataset	Cora	Citeseer	Pubmed
CRF-GCN-Gaussian	0.828	0.718	0.790
CRF-GCN-Gaussian-const	0.813	0.698	0.791
CRF-GCN-NN	0.825	0.721	0.792
CRF-GCN-NN-const	0.811	0.700	0.788

to compute and optimize so that it can be inserted into existing graph convolutional neural networks to improve their performance. The extensive experimental results have verified the effectiveness of our proposed method.

REFERENCES

[1] Rie K Ando and Tong Zhang. 2007. Learning on graph with Laplacian regularization. In *Advances in neural information processing systems*. 25–32.

[2] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1993–2001.

[3] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research* 7, Nov (2006), 2399–2434.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096* (2018).

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[6] Miguel Carreira-Perpinan and Weiran Wang. 2014. Distributed optimization of deeply nested systems. In *Artificial Intelligence and Statistics*. 10–19.

[7] Dapeng Chen, Dan Xu, Hongsheng Li, Nicu Sebe, and Xiaogang Wang. 2018. Group Consistent Similarity Learning via Deep CRF for Person Re-Identification. In *CVPR*.

[8] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247* (2018).

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2018. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2018), 834–848.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.

[11] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*. 2224–2232.

[12] Hongchang Gao and Heng Huang. 2018. Deep Attributed Network Embedding.. In *IJCAI*.

[13] Hongchang Gao and Heng Huang. 2018. Joint Generative Moment-Matching Network for Learning Structural Latent Code.. In *IJCAI*.

[14] Hongchang Gao and Heng Huang. 2018. Self-paced network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1406–1415.

[15] Hongchang Gao, Jian Pei, and Heng Huang. 2019. ProGAN: Network Embedding via Proximity Generative Adversarial Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.

[16] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[19] Xiaofei He and Partha Niyogi. 2004. Locality preserving projections. In *Advances in neural information processing systems*. 153–160.

[20] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

[21] Yedid Hoshen. 2017. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*. 2701–2711.

[22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[23] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[24] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Combining Neural Networks with Personalized PageRank for Classification on Graphs. *arXiv* (2018).

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[26] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[27] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. 2018. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*. 537–546.

[28] Xiaodan Liang, Zhiting Hu, Hao Zhang, Liang Lin, and Eric P Xing. 2018. Symbolic graph reasoning meets convolutions. In *Advances in Neural Information Processing Systems*. 1858–1868.

[29] Qing Lu and Lise Getoor. 2003. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 496–503.

[30] Tengfei Ma, Cao Xiao, Junyuan Shang, and Jimeng Sun. 2018. CGNF: Conditional Graph Neural Fields. (2018).

[31] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[32] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proc. CVPR*, Vol. 1. 3.

[33] Medhini Narasimhan, Svetlana Lazebnik, and Alexander Schwing. 2018. Out of the box: Reasoning with graph convolution nets for factual visual question answering. In *Advances in Neural Information Processing Systems*. 2659–2670.

[34] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International conference on machine learning*. 2014–2023.

[35] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[36] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

[37] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290, 5500 (2000), 2323–2326.

[38] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*. 4967–4976.

[39] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[42] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. 2012. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*. Springer, 639–655.

[43] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861* (2016).

[44] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. *arXiv preprint arXiv:1806.01973* (2018).

[45] Huishuai Zhang, Wei Chen, and Tie-Yan Liu. 2018. On the local Hessian in back-propagation. *NIPS*.

[46] Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185* (2018).

[47] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.

[48] Chenyi Zhuang and Qiang Ma. 2018. Dual Graph Convolutional Networks for Graph-Based Semi-Supervised Classification. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 499–508.