SDG: A Simplified and Dynamic Graph Neural Network

Dongqi Fu
University of Illinois at Urbana-Champaign
dongqif2@illinois.edu

Jingrui He
University of Illinois at Urbana-Champaign
jingrui@illinois.edu

ABSTRACT

Graph Neural Networks (GNNs) have achieved state-of-the-art performance in many high-impact applications such as fraud detection, information retrieval, and recommender systems due to their powerful representation learning capabilities. Some nascent efforts have been concentrated on simplifying the structures of GNN models, in order to reduce the computational complexity. However, the dynamic nature of these applications requires GNN structures to be evolving over time, which has been largely overlooked so far. To bridge this gap, in this paper, we propose a simplified and dynamic graph neural network model, called SDG. It is efficient, effective, and provides interpretable predictions. In particular, in SDG, we replace the traditional message-passing mechanism of GNNs with the designed dynamic propagation scheme based on the personalized PageRank tracking process. We conduct extensive experiments and ablation studies to demonstrate the effectiveness and efficiency of our proposed SDG. We also design a case study on fake news detection to show the interpretability of SDG.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Neural networks**; **Learning latent representations**;

KEYWORDS

Graph Neural Networks, Scalability, Interpretability

ACM Reference Format:
Dongqi Fu and Jingrui He. 2021. SDG: A Simplified and Dynamic Graph Neural Network. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3463059

1 INTRODUCTION

Graphs are ubiquitous data structures nowadays for representing rich information regarding node interactions and connections. Combined with the powerful representation learning capabilities of deep neural network models, Graph Neural Networks (GNNs) [8, 11, 22] have achieved state-of-the-art performance in many high-impact applications, such as fraud detection [14], information retrieval [16], and recommender systems [23, 26].

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463059

Following the success of GNNs, many nascent efforts concentrate on scaling GNNs during the information propagation process among nodes, which aims to reduce the computational complexity of the training stage (i.e., to reduce the training time or the number of parameters) but maintain the performance in the mean time [2, 3, 10, 12, 24]. For example, in [3], the authors integrate the importance sampling scheme into the message-passing mechanism to sample subgraphs in each layer for scaling the structures of GNNs. Inspired by [25], the authors in [12] directly replace the message-passing mechanism with the random walk stationary distribution among nodes to avoid stacking layers in GNNs. However, when applying GNNs to the real-world complex settings, in addition to scaling their structures, another critical aspect has been largely overlooked, i.e., evolving the structures of GNNs. Compared with static neural network models with fixed structures and parameters, dynamic models have the potential of simultaneously enjoying compatibility and interpretability due to the adaptation of their structures or parameters to different inputs [9]. Therefore, we make the first attempt to simplify and dynamize the structures of GNNs for the scalability and interpretability, which is different from the graph representation learning models learning the evolution pattern [15] or persistent pattern [5] of dynamic graphs.

To this end, we propose a simplified and dynamic graph neural network model in this paper, called SDG. In the proposed SDG, we design the dynamic propagation scheme based on the personalized PageRank tracking process without the explicit message-passing to transfer multi-hop neighbourhood information. The key idea behind this modification is that the influence of other nodes on a selected node in the graph through a k-layer GNN model is proportional to the k-step random walk distribution starting from that selected seed node [12, 25]. The advantage of the dynamic propagation scheme is that when the structure of input graph changes, the stationary distribution of random walks could be tracked in a fast and accurate manner instead of resolving from scratch. Hence, we can leverage this property to (1) not only achieve efficient fine-tuning for node-level and graph-level classification on changed graphs, (2) but also investigate the influence of a certain node on the node-level and graph-level label prediction results by masking that node in the proposed dynamic propagation scheme. Then we design extensive experiments and ablation studies to demonstrate the effectiveness and efficiency of our proposed SDG model in node classification tasks on changed graphs, compared with baseline algorithms. We also design a case study of fake news detection based on the pre-trained word embeddings and constructed news article word graphs, in order to investigate the influence of a certain word on the detection result of a news article.

2 PRELIMINARIES

In this section, we first introduce the notation used in this paper, and explain the basic theory behind our proposed model.

We use bold capital letters to denote matrices (e.g., $\boldsymbol{A}$) and bold lower-case letters to denote column vectors (e.g., $\boldsymbol{a}$). We follow the convention of Matlab for matrix indexing (e.g., $A(i,:)$ denotes the i-th row of $\boldsymbol{A}$). Given an undirected and unweighted graph $G = (V, E)$ with nodes V and edges E, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ denotes the adjacency matrix with self-loops, $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ denotes the degree matrix with self-loops, $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ denotes the node feature matrix, and $\boldsymbol{Y} \in \mathbb{R}^{n\times c}$ denotes the node label matrix, where n is the number of nodes, d is the dimension of node features, and c is the dimension of node labels. Moreover, when the input graph G changes with inserted and deleted edges ΔE, we use G' to denote the new graph. For the dimension consistency of matrices during the proposed dynamic propagation scheme, we consider the number of nodes to be fixed, i.e., an inserted (or deleted) node is regarded as a previous (or existing) dangling node [21].

In [25], authors have shown that the influence of a node V_j on a node V_i through a k-layer GNN model [8, 11] is proportional to the k-step random walk distribution on node V_j starting from the seed node V_i. Also, when $k \to \infty$, the random walk distribution converges to the stationary distribution. That paves the way for improving the scalability of GNNs by replacing the traditional message-passing mechanism with the stationary distribution (i.e., personalized PageRank) to propagate information among nodes [2, 12], where the training time and the number of parameters are reduced for avoiding stacking layers in neural networks.

3 PROPOSED MODEL

In this section, we first introduce the overall framework of our proposed SDG graph neural network model, and then we illustrate each component of SDG in a systematic way.

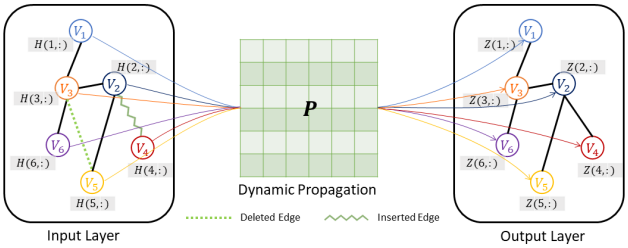

Figure 1: Structure of SDG Graph Neural Network.

3.1 Overview of SDG

The proposed SDG graph neural network model has an input layer and an output layer as shown in Figure 1. To dynamically use the stationary distribution to propagate the neighborhood information among nodes instead of stacking layers, we propose the dynamic propagation scheme in our SDG graph neural network, which is expressed as follows.

$$\boldsymbol{Z} = \text{softmax}(\boldsymbol{PH}) \tag{1}$$

where $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ is the dynamic propagation matrix, $\boldsymbol{H} \in \mathbb{R}^{n\times c}$ is the hidden node feature matrix extracted from the input node features $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ through the model-agnostic neural network f_θ, i.e., $\boldsymbol{H} = f_\theta(\boldsymbol{X})$, and $\boldsymbol{Z} \in \mathbb{R}^{n\times c}$ denotes the predicted label matrix.

Next we introduce how to form the dynamic propagation matrix $\boldsymbol{P}$ for the changed graph structure in Subsection 3.2, and how to design the model-agnostic neural network f_θ to extract the hidden node feature matrix $\boldsymbol{H}$ from the input node feature matrix $\boldsymbol{X}$ in Subsection 3.3.

Finally, the loss function of SDG is defined as follows.

$$\mathcal{L}_{\text{SDG}} = -\sum_{i}^{n}\sum_{j}^{c} Y(i,j)\ln Z(i,j) \tag{2}$$

where $\boldsymbol{Y} \in \mathbb{R}^{n\times c}$ and $\boldsymbol{Z} \in \mathbb{R}^{n\times c}$ are the ground truth and model output respectively.

3.2 Dynamic Propagation Scheme

The proposed dynamic propagation scheme is realized by tracking the dynamic propagation matrix $\boldsymbol{P}$, where each row $P(i,:)$ is occupied by the stationary distribution of random walks starting from that node V_i. To be specific, $P(i,:)$ can be fast tracked instead of solving from scratch when the graph topology changes from $\boldsymbol{A}$ to $\boldsymbol{A}'$. And the computation of each row is independent, which implies that the tracking process from $\boldsymbol{P}$ to $\boldsymbol{P}'$ can be parallelized. We first introduce the tracking process of $\boldsymbol{P}'$, and then analyze its time complexity and error bound.

As each row $P(i,:)$ encodes the stationary distribution of random walks starting from node V_i, it can be expressed as follows.

$$P(i,:)^\top = \alpha \boldsymbol{M} P(i,:)^\top + (1-\alpha)\boldsymbol{r} \tag{3}$$

where $\boldsymbol{M} = \boldsymbol{A}\boldsymbol{D}^{-1} \in \mathbb{R}^{n\times n}$ denotes the column-stochastic transition matrix, $\alpha \in [0,1]$ denotes the teleportation probability, and $\boldsymbol{r} \in \mathbb{R}^n$ denotes the personalized vector with $r(i) = 1$ and other entries equal to 0. For the clear notation, we omit the transpose notation $\top$ for $P(i,:)$ in the following part.

When the graph topology changes from $\boldsymbol{M}$ to $\boldsymbol{M}'$ with the updated edge ΔE, the stationary distribution needs to be updated. To obtain $\boldsymbol{P}'$, we need update each row of $\boldsymbol{P}$. The core idea is to push out the previous probability distribution score from the changed part to the residual part of the graph [6, 27], and then add the pushed out distribution back to the previous distribution in order to finally obtain the new distribution. The tracking process can be described as follows.

$$P(i,:)_{pushout} = \alpha(\boldsymbol{M}' - \boldsymbol{M})P(i,:) \tag{4}$$

and

$$P(i,:)' = P(i,:) + \sum_{k=0}^{\infty}(\alpha\boldsymbol{M}')^k\, P(i,:)_{pushout} \tag{5}$$

where $P(i,:)_{pushout}$ denotes the distribution score that needs to be pushed out on the residual graph due to the updated edges ΔE, and $P(i,:)'$ denotes the tracked new distribution.

This push out process can be proved to converge to the exact stationary distribution of the new graph through sufficient cumulative power iterations [27, 28], i.e., $k \to \infty$. With this push out process, we can update each row of dynamic propagation matrix $\boldsymbol{P}$ to further track $\boldsymbol{P}'$. Next we analyze the time complexity and accuracy of approximately tracking the whole dynamic propagation matrix $\boldsymbol{P}'$ with a threshold ϵ as follows.

Theorem 1. *Given the updated matrix $\boldsymbol{M}'$, the threshold ϵ, and q distributed machines, each machine at most costs $O(\frac{mn}{q}\log_\alpha(\frac{\epsilon}{2}))$ time to obtain the tracked dynamic propagation matrix $\boldsymbol{P}'$ with each row bounded by $\frac{\epsilon}{1-\alpha}$ error compared with the stationary distribution*

in terms of ℓ_1-norm, where m is the number of non-zero entries of $\boldsymbol{M'}$, and n is the number of nodes in the graph.

Proof. With the threshold ϵ, if $\|(\alpha \boldsymbol{M'})^k \ P(i,:)_{pushout}\|_1 \leq \epsilon$ then we stop the iteration of Eq. 5. Since $\boldsymbol{M'}$ is a column-stochastic matrix, then $\|\boldsymbol{M'}\|_1 = 1$. Thus, the number of iterations k satisfies $k \leq \log_\alpha(\frac{\epsilon}{\|P(i,:)_{pushout}\|_1})$; and $P(i,:)_{pushout} = \alpha(\boldsymbol{M'} - \boldsymbol{M})P(i,:)$, then $\|P(i,:)_{pushout}\|_1 \leq 2\alpha$. The computation of each row is independent, such that each distributed machine at most costs $O(\frac{n}{q} m \log_\alpha(\frac{\epsilon}{2}))$ complexity. Also, denoting the early stop at the s-th iteration, then the remaining tracking error is denoted as $\| \sum_{k=s}^{\infty} (\alpha \boldsymbol{M'})^k \ P(i,:)_{pushout}\|_1$, which can be proved less than $\frac{\epsilon}{1-\alpha}$ just by replacing $P(i,:)_{pushout}$ with $\alpha(\boldsymbol{M'} - \boldsymbol{M})P(i,:)$ and setting $k = log_\alpha(\frac{\epsilon}{2\alpha})$. □

Thus, tracking the whole dynamic propagation matrix $\boldsymbol{P'}$ costs $O(\frac{mn}{q} \log_\alpha(\frac{\epsilon}{2}))$ time complexity with the error bound $\frac{n\epsilon}{1-\alpha}$ to avoid solving it from scratch by matrix inversion with $O(n^3)$ [12].

3.3 Model-Agnostic Neural Networks

The intuition of applying the model-agnostic neural network f_θ is to extract the qualified hidden node features $\boldsymbol{H}$ from the input node features $\boldsymbol{X}$. f_θ can take a variety of forms, such as CNNs [7]. Without loss of generality, we use a linear multilayer perceptron as f_θ, to avoid increasing too much training complexity. Note that f_θ operates on each input node independently, which implies that extracting hidden node features could also be paralleled as follows.

$$H(i,;) = f_\theta(X(i,:)) \tag{6}$$

where $X(i,:)$ denotes the input node features of node V_i, and $H(i,:)$ denotes the hidden node feature of node V_i extracted by f_θ.

Since the dynamic propagation $\boldsymbol{P}$ can be pre-computed independently ahead of the training process of f_θ, we separate the information propagation from the node feature extraction. It makes the scalability and interpretability of SDG neural network possible, when the input graph structure or the input node feature changes.

Scalability of SDG. When the structure of the input graph changes (i.e., from $\boldsymbol{A}$ to $\boldsymbol{A'}$) and/or the input node features change (i.e., from $\boldsymbol{X}$ to $\boldsymbol{X'}$), the dynamic propagation matrix $\boldsymbol{P'}$ can be fast obtained in parallel as mentioned above. In this case, f_θ only needs to be fine-tuned to produce $\boldsymbol{H'}$, starting from the previously well-trained parameters.

Interpretability of SDG. To investigate the influence of a certain node V_i on the final prediction, SDG enables two types of methods, i.e., from the node feature view or from the graph structure view. First, we remove the hidden node features $H(i,:) = \mathbf{0}$ and keep all remaining parts to see the change of the prediction results on other nodes. Second, we mask (i.e., delete) certain edges related to V_i on the graph, fast track the new stationary distribution, and keep all remaining parts to see the change of the final predictions.

3.4 Optimization

We summarize the training process of SDG in Algorithm 1. In Steps 1-4, the initial inputs could be obtained from existing static algorithms like [2, 12]. Then in Steps 5-9, SDG incrementally tracks the dynamic propagation matrix and fine-tunes the model-agnostic neural network parameters if any change happens on the graph structure level and/or the node feature level.

Algorithm 1 Stochastic Training Procedure for SDG

Input:
Graph G with adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, label matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times c}$, and any possible updates $\boldsymbol{A'} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{X'} \in \mathbb{R}^{n \times d}$.

Output:
Predictions $\boldsymbol{Z} \in \mathbb{R}^{n \times c}$ and updated predictions $\boldsymbol{Z'} \in \mathbb{R}^{n \times c}$

1: Compute the initial dynamic propagation matrix $\boldsymbol{P}$.
2: **while** not converge **do**
3: Train f_θ through Eq. 1 and Eq. 2 to obtain qualified $\boldsymbol{H}$ and $\boldsymbol{Z}$.
4: **end while**
5: **if** graph structure and/or node features change **then**
6: Update $\boldsymbol{P}$ into $\boldsymbol{P'}$ through Eq. 4 and Eq. 5.
7: **end if**
8: **while** not converge **do**
9: Fine-tune f_θ on any new matrices $\boldsymbol{A'}$ and/or $\boldsymbol{X'}$.
10: **end while**

4 EXPERIMENTS

In this section, we first introduce the data sets used for effectiveness comparison and efficiency analysis with state-of-the-art baseline algorithms in node classification tasks[1]. Then, to show the interpretability of our SDG model, we design the fake news detection case study by using the proposed SDG graph neural network model to classify news article word graphs, and to evaluate the influence of each word on the prediction result.

Table 1: Dataset Statistics

Dataset	Classes	Nodes	Edges	Label Rate
Citeseer	6	2,110	3,668	0.036
Cora-ML	7	2,810	7,981	0.047
PubMed	3	19,717	44,324	0.003

4.1 Data Sets

We use three real-world citation graphs (i.e., Citeseer [20], Cora-ML [1, 17], and PubMed [19]) to design experiments for the text classification problem. In the citation graph, each node represents a paper, and each edge represents the citation relationship between two papers. We leverage the largest connected component of each graph during the experiment, and each node feature is extracted by a bag-of-words representation of that paper's abstract. The statistics of the three data sets are shown in Table 1.

4.2 Baseline Algorithms

PPNP and APPNP are two simplified graph neural network models [12] with state-of-the-art effective and efficient performance in many graph mining tasks [2, 12]. Also, we include a variate of our SDG called SDG-S for the ablation study, which removes the dynamic propagation scheme, i.e., k always equals to 0 in Eq. 5. With this variate, we can investigate the capability of the proposed dynamic propagation scheme in terms of improving effectiveness and efficiency.

[1] https://github.com/DongqiFu/SDG

Table 2: Effectiveness and Efficiency Comparison

Methods	Citeseer		Cora-ML		PubMed	
	Accuracy (%)	Time Consumption (s)	Accuracy (%)	Time Consumption (s)	Accuracy (%)	Time Consumption (s)
PPNP	74.07±0.53	10.89±0.91	84.40±0.18	19.81±1.65	84.03±0.32	109.75±7.68
APPNP	73.93±0.30	22.80±1.69	84.63±0.34	49.70±6.18	83.73±0.21	39.91±4.29
SDG-S	74.10±0.30	6.65±0.69	84.60±0.28	8.51±3.26	84.10±0.28	12.74±5.05
SDG	74.17±0.39	7.11±0.93	84.87±0.68	5.89±3.22	84.70±0.59	16.06±6.45

4.3 Effectiveness and Efficiency Comparison

Due to the difference between static and dynamic graph neural network models, we mask 1% − 3% edges of the whole graph to form the graph G and add back those edges to form the graph G'. After setting $\alpha = 0.9$ and making all algorithms converge at the same error threshold, we report the average node classification accuracy and time consumption on graph G' of all baselines. Note that PPNP and APPNP are trained solely on graph G', SDG and SDG-S are trained on graph G and fine-tuned on graph G'. In Table 2, it can be observed that SDG and SDG-S could fast provide competitive performance for node classifications. To be specific, in PubMed, SDG achieves 84.70% accuracy that is 0.79% higher than the third best (PPNP). An intuitive explanation is that the tracked stationary distribution of SDG is more suitable for the parameters of the model-agnostic neural network to classify nodes. Moreover, SDG-S also provides the acceptable accuracy in a fast manner for avoiding the tracking process only with the outdated stationary distribution. A possible explanation is that the topology update between two graphs G and G' is relatively small as compared to the whole graph, and the model-agnostic neural network bridges the insufficient tracking with its adequate representative abilities. To verify this explanation, we design the following parameter sensitivity analysis.

4.4 Parameter Sensitivity

To verify the above mentioned explanation, we change the value of two important parameters, i.e., the teleportation probability α and the number of tracking iterations k, and to see how the performance changes accordingly. In Figure 2, as the number of tracking iterations increases, the classification accuracy increases but only to a small extent. This observation suggests that the model-agnostic neural network bridges the gap between approximated tracked stationary distribution and the real stationary distribution, which means the classification accuracy is not very sensitive to k when the graph structure does not change too much. Therefore, the proposed SDG model can be readily applied to interpret the prediction result, i.e., when only a small portion of the input graph is masked, a small k can also provide comparable performance.

4.5 Case Study

Disinformation dissemination has a huge negative impact on our society, and many factors may hinder the identification of disinformation. For example, messages on the social network are usually short and fast propagated, which requires that the predictive model should have efficient training process and adaptable structures for the evolving environment. Hence, we design the fake news case study with the proposed SDG model and a fake news data set [13].

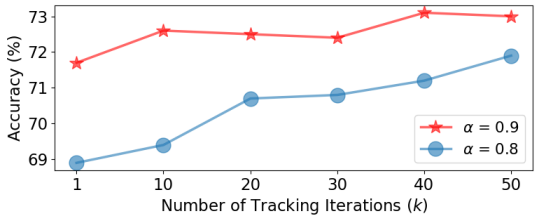

Figure 2: Parameter Sensitivity of SDG on Citeseer Dataset.

Our objective is news article word graph classification. To build a word graph (i.e., Figure 3), a word stands for a node, an edge is established if two words co-occur in a window of q text units [18]. Then, each node has a word embedding vector from the pre-trained Bert model [4], and the word graph feature aggregates all node embeddings through an average pooling layer. Also, each word graph associates with a label indicating fake news or not. In Figure 3, when we mask nodes "drinking water" and "eliminates COVID-19 virus" respectively, that fake news is detected with probability 99.29% and 62.95%, which suggests that "eliminates COVID-19 virus" is a key indicator to detect that fake news article.

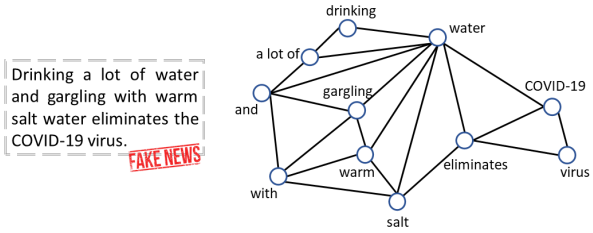

Figure 3: A Fake News Article and its Word Graph.

5 CONCLUSION

In this paper, we propose a simplified and dynamic graph neural network model, named SDG, which replaces the traditional message-passing mechanism with the designed dynamic propagation scheme. We design extensive experiments on real-world graphs to demonstrate the effectiveness, scalability, and interpretability of SDG, in comparison with state-of-the-art techniques.

ACKNOWLEDGEMENT

This work is supported by National Science Foundation under Award No. IIS-1947203 and IIS-2002540, and the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-03-03. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR 2018*.

[2] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling Graph Neural Networks with Approximate PageRank. In *KDD 2020*.

[3] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR 2018*.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019*.

[5] Dongqi Fu, Zhe Xu, Bo Li, Hanghang Tong, and Jingrui He. 2020. A View-Adversarial Framework for Multi-View Network Embedding. In *CIKM 2020*.

[6] Dongqi Fu, Dawei Zhou, and Jingrui He. 2020. Local Motif Clustering on Time-Evolving Graphs. In *KDD 2020*.

[7] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. 2015. Recent Advances in Convolutional Neural Networks. *CoRR* (2015).

[8] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS 2017*.

[9] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. 2021. Dynamic Neural Networks: A Survey. *CoRR* (2021).

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR 2020*.

[11] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR 2017*.

[12] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR 2019*.

[13] Nayeon Lee, Yejin Bang, Andrea Madotto, and Pascale Fung. 2020. Misinformation Has High Perplexity. *CoRR* (2020).

[14] Zhiwei Liu, Yingtong Dou, Philip S. Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. In *SIGIR 2020*.

[15] Yao Ma, Ziyi Guo, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming Graph Neural Networks. In *SIGIR 2020*.

[16] Kelong Mao, Xi Xiao, Jieming Zhu, Biao Lu, Ruiming Tang, and Xiuqiang He. 2020. Item Tagging for Information Retrieval: A Tripartite Graph Neural Network based Approach. In *SIGIR 2020*.

[17] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the Construction of Internet Portals with Machine Learning. *Inf. Retr.* (2000).

[18] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *EMNLP 2004*.

[19] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. 2012. Query-driven active surveying for collective classification. In *MLG 2012*.

[20] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Mag.* (2008).

[21] Hanghang Tong, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. 2008. Proximity Tracking on Time-Evolving Bipartite Graphs. In *SDM 2008*.

[22] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR 2018*.

[23] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR 2019*.

[24] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML 2019*.

[25] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML 2018*.

[26] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD 2018*.

[27] Minji Yoon, Woojeong Jin, and U Kang. 2018. Fast and Accurate Random Walk with Restart on Dynamic Graphs with Guarantees. In *WWW 2018*.

[28] Minji Yoon, Jinhong Jung, and U Kang. 2018. TPA: Fast, Scalable, and Accurate Method for Approximate Random Walk with Restart on Billion Scale Graphs. In *ICDE 2018*.