Research

Social behavior prediction with graph U-Net+

Zhiyue Yan¹ · Wenming Cao¹ · Jianhua Ji²

Received: 29 June 2021 / Accepted: 3 August 2021

Published online: 06 September 2021

© The Author(s) 2021 [OPEN](#)

Abstract

We focus on the problem of predicting social media user's future behavior and consider it as a graph node binary classification task. Existing works use graph representation learning methods to give each node an embedding vector, then update the node representations by designing different information passing and aggregation mechanisms, like GCN or GAT methods. In this paper, we follow the fact that social media users have influence on their neighbor area, and extract subgraph structures from real-world social networks. We propose an encoder–decoder architecture based on graph U-Net, known as the graph U-Net+. In order to improve the feature extraction capability in convolutional process and eliminate the effect of over-smoothing problem, we introduce the bilinear information aggregator and NodeNorm normalization approaches into both encoding and decoding blocks. We reuse four datasets from DeepInf and extensive experimental results demonstrate that our methods achieve better performance than previous models.

Keywords Social behavior prediction · Graph representation learning · Graph U-Net · Node classification

1 Introduction

With the explosive appearance of social media such as Twitter, WeChat, and Weibo, many people have joined the Internet and thus formed some huge social network neighborhoods. As shown in Fig. 1, these social media allow users to directly view the contents of all their online friends, and support users to further improve the breadth and depth of information propagation through online interactions, such as subscribing, sharing, and retweeting. During the process of user interaction, user-level social influence naturally appears and imperceptively “interferes” with the behavior and judgment of each user [1–3]. The term social influence refers to the impact that user's opinions, decisions, or behaviors may be affected by their social neighbors or some influential people. For example, we choose whether to watch movies based on suggestion of our closest friend. For another instance, a person may decide whether to believe a piece of political news based on what an authoritative official says. Therefore, collecting and analyzing users friendship network and history data makes it possible for governments or companies to predict users' future behavior, so as to determine whether he or she will engage in certain activities such as voting or purchasing. The resulting social user behavior prediction task has important value in many application fields, such as online advertising [4, 5], recommendation systems [6–8], viral influence prediction in marketing [9–11], rumor spreading [12], and even presidential elections [13].

Indeed, extensive work has been done on social behavior prediction in the literature [2, 14–16]. For example, Li et al. [16] proposed an end-to-end predictor, DeepCas, which studies the diffusion cascade of social influence by

✉ Jianhua Ji, jjh@szu.edu.cn; Zhiyue Yan, 1810263026@email.szu.edu.cn; Wenming Cao, wmcao@szu.edu.cn | ¹College of Electronic and Information Engineering, Shenzhen University, Room N307, Shenzhen, China. ²College of Electronic and Information Engineering, Shenzhen University, Room N808, Shenzhen, China.

Fig. 1 An illustration of social network neighborhood. In this social network, everyone can influence others or be influenced by online neighbors. The yellow lines represent the interaction or friendship link between different social media users

incorporating bidirectional-GRU and Node2Vec to achieve effective prediction of user behavior. With the recent success of graph deep learning, some works focus on the graph convolutional network [17, 18], network embedding which not only reduce dimension but also put more emphasis on retaining network properties as much as possible [19–22], and graph attention mechanism [23, 24]. For example, Qiu et al. [25] aim to predict the behavior status of a user given the behavior statuses of his or her local structure information, thus they propose a deep learning framework, DeepInf which leverage both graph convolution and attention techniques, to represent both influence dynamics and network structures into a latent space. Another recent research based on graph deep learning is Personalized-DeepInf [26], it integrates teleport probability from pagerank algorithm into GCN [17] model and improve the performance of PPNP/APPNP [27] algorithm. Although these works have made some pioneering progress, their models still suffer from some problems, including over-smoothing, lack of more effective methods on information aggregation and feature extraction, and simple shallow architecture.

The recent survey [28] pointed out that the social network analysis and research can also be utilized in anomaly detection, which has received attracted more attention on social science, computer science, chemistry, and biology. Anomalies represent rare observations (e.g., data records, messages or events) that are deviating significantly from others. Anomaly detection, which aims to identify these rare observations, is among the most vital tasks and has shown its power in preventing detrimental events, such as financial fraud, network intrusion, and social spam, from happening. Graphs have been prevalently used to preserve the structural information, and this raises the graph anomaly detection problem—identifying anomalous graph objects (i.e., nodes, edges and sub-graphs). In the research field of social network behavior prediction, graph anomaly detection with deep learning has received intensified studies. For example, abnormal users appear, potential hacker users destroy the network.

Inspired by the success of U-Net [29] framework in CNN, Gao et al. [30] develop another encoder–decoder U-Net-like architecture named “graph U-Net” which bridge the gap of graph pooling and unpooling operations for graph data. Images can be treated as a grid graph data, so there is a natural correspondence between the node classification task and the pixel-level prediction task of semantic segmentation on images. To be specific, the goal of both tasks is to make categorical predictions for each input unit which corresponds a pixel or a node. Graph U-Net is an advanced and popular network architecture in graph deep learning, it has strong capability of feature extraction and node representation learning. Follow this work, we propose a new architecture called graph U-Net+ to solve the social behavior prediction problem.

In this paper, we make improvements on the original graph U-Net from the blocks designing to the entire network structure building. We stack several graph U-Nets at different depths as the main body of graph U-Net+. In order to ensure the back-propagation and gradient descent algorithm works, we not only change the original skip-connection way of graph U-Net from residual to dense, but also add a deep supervision mechanism to the loss function with four outputs of different encoders. The graph U-Net+ extends the information aggregation approach by combining GCNII [31] and bilinear information aggregator [32] for all encoding and decoding blocks. In addition, we also introduce a novel graph normalization technique named NodeNorm [33], so as to eliminate the over-smoothing problem with

more efficiency. Compared to graph U-Net, the experimental result show that graph U-Net+ has better performance due to its effective architecture and powerful information aggregation mechanism. Moreover, we implement many ablation experiments to test effectiveness of different modules. In the end, we give the pseudocode of the entire program codes running for readers to better understand how we train, validate and test.

The rest of this paper is organized as follows: Sect. 2 introduces some related works. Section 3 gives the details of our proposed architecture graph U-Net+. In Sects. 4 and 5, we conduct extensive experiments on four illustrated datasets and analyze the results. Finally, we draw a conclusion in Sect. 6.

2 Related work

In this section, we will introduce some related work from two aspects: graph representation learning (information aggregation layer) and graph U-Net architecture.

Recently, there has been many works on graph representation learning. As a general framework, MPNN [34] abstracts the commonalities of existing models, and thinks of them as an unified process of information propagation and aggregation. Some studies applied this framework to graph node classification tasks. Inspired by the ChebNet [35] and first order graph Laplacian methods, Kipf and Welling proposed graph convolutional network (GCN) [17] which achieved promising performance. The information aggregation layer in GCN is defined as:

$$X_{\ell+1} = \sigma \left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} X_{\ell} W_{\ell} \right) \quad (1)$$

where $\hat{A} = A + I$ is used to add self-loops in the input adjacency matrix A , X_{ℓ} is the node feature matrix which each row represents a feature vector of one node, W_{ℓ} is a trainable weight matrix that applies a linear transformation to feature vectors of layer ℓ . GCN uses the diagonal node degree matrix $\hat{D}$ to normalize $\hat{A}$, and essentially perform information aggregation and transformation on node features like what MPNN does. In spatial domain, the GraphSage [18] samples a fixed number of neighboring nodes to simulate the information aggregation process of first-order approximate GCN after stacking multiple layers. GAT [23] uses self-attention mechanisms to assign different weights for neighboring nodes.

In addition to graph representation learning, some studies tried to extend pooling operations from CNN to graph domain. The ChebNet [35] tried to fix indices of nodes before applying 1-D pooling operations by using binary tree indexing for graph coarsening. Simonovsky and Komodakis [36] used deterministic graph clustering algorithm to determine pooling patterns. Ying et al. [37] proposed DiffPool which used an assignment matrix to achieve pooling by assigning nodes to different clusters of the next layer. In graph U-Net [30], Gao and Ji proposed novel graph pooling (gPool) operation, they set a global projection vector, and the projection of node feature vectors is regarded as the importance degree. Then, the importance degree of nodes is used to make a contraction transformation of node features, which further enhances the gradient learning of nodes with higher importance degree. The gPool layer adopts the method of discarding nodes layer by layer to improve the efficiency of information aggregation of long-distance nodes. This is an important implementation of pooling operations on graph-structured data based on TopK scheme. Correspondingly, an unpooling operation named gUnpool is proposed in graph U-Net, which restores the graph to its original structure with the help of locations of nodes selected in the corresponding gPool layer.

The pooling for downsampling and unpooling for upsampling operations are the key to the U-Net architecture. Based on the gPool and gUnpool layers mentioned above, Gao and Ji developed the graph U-Net architecture [30], which allows high-level feature encoding and decoding for network embedding. In the encoder part, each encoding block contains a gPool layer followed by a GCN layer. gPool layers reduce the size of feature map to encode high-order features for coarsened graph, while GCN layers passes and aggregates information from the first-order of each node. In the decoder part, there is the same number of decoding blocks as in the encoder part. Each decoder block consists of a gUnpool layer and a GCN layer, which gUnpool layer restores the graph into its original structure with higher resolution. Besides, there are skip-connections like residual connections between corresponding blocks of encoder and decoder parts, which transmit spatial information from encoding blocks to decoding blocks for better performance. Figure 2 provides the architecture of graph U-Net and instructions of gPool/gUnpool layers. Moreover, the graph U-Net designs two tricks to improve the performance. One trick is to augment the connectivity of graph by graph power operation, since some less important nodes and related edges are removed through gPool layers, the nodes in the coarsened graph might become isolated that may influence the information propagation and aggregation in subsequent GCN layers. The graph U-Net uses graph power $\mathbb{G}^2 : A^2 = A^{\ell} A^{\ell}$ to build links between

Fig. 2 An illustration of the graph U-Net architecture and its encoder–decoder parts. **a** In this example, each node in the input graph has 2 features for node classification task. It has 2 encoding/decoding blocks and 2 skip-connections. **b** The graph pooling layer considers the situation of a graph with 4 nodes, and each node has 5 features. After pool with 0.5 pooling ratio, the coarsened graph has 2 nodes. **c** For example, a graph with 7 nodes is down-sampled, resulting in a coarsened graph with 4 nodes and position information of selected nodes. The corresponding gUnpool layer uses the position information to reconstruct the original graph structure by distributing empty feature vectors to unselected nodes

nodes whose distances are at most 2 hops, and select the nodes $A^{\ell+1} = A^2(\text{idx}, \text{idx})$ through the TopK ranking result of gPool layer as shown in Fig. 2b. The other one is the improved GCN layer, which means the adjacency matrix before normalization is computed as $\hat{A} = A + 2I$ by commonly imposing larger weights on self loops in the graph processing. Although the graph U-Net achieves the encoder–decoder architecture with two small operation tricks on graph data, it still has two limitations. First, the optimal depth of an encoder–decoder network can vary due to applications, depending on the task difficulty and complexity. This uncertain situation is inefficient from a deployment perspective, because these networks do not share a common encoder. Second, the design of skip connections used in an encoder–decoder network is unnecessarily restrictive, demanding the fusion of the same-scale encoder and decoder feature maps. The graph U-Net handicapped by unnecessarily restrictive skip connections where only the same-scale feature maps from the encoder and decoder can be fused. While the same-scale feature maps from the decoder and encoder networks are semantically dissimilar. Thus, there is a need for a redesigned skip connection way which presents feature maps of varying scales at a decoder node, allowing the aggregation layer to decide how various feature maps carried along the skip connections should be fused with the decoder feature maps, such as the dense connection way.

In addition to the graph tasks for classification described above, graph representation can also be applied to social network analysis and community detection and recommendation tasks [38, 39]. A community reveals the characteristics of its members that are different from other communities in the network and detection of connected communities is of great significance in network analysis. The community detection applications usually contains community deception, community search, recommendation systems and online social network analysis.

Fig. 3 We compress the four graph U-Nets of different depths that maintain the original skip-connection way together, and find that the red region cannot be trained through back propagation

3 Graph U-Net+

In this section, we use graph U-Net for reference and propose the graph U-Net+ architecture for social behavior prediction (node binary classification) task. Based on this novel U-Net-like architecture, we introduce some improvements, including improved encoding/decoding blocks, better skip-connection way, and deep supervision mechanism.

3.1 Graph U-Net+ architecture

It is well-known that encoder–decoder structure networks like graph U-Net has achieved promising performance on node-wise prediction tasks, since they can encode and decode high-level features while maintaining local spatial information. The down-sampling operation gPool is adopted to increase the robustness of the model, such as subtle changes in the graph structure and signal noises, and reduce the risk of over-fitting. The size of the graph is compressed to reduce the computation. In this way, the graph U-Net cannot only capture the features of shallow layers, but also capture the deep layer features caused by the gradual downward expansion of receptive field and graph convolution.

In order to make full use of the respective advantages of graph U-Net networks at different depths to capture features at different levels, we stack all graph U-Net networks with depths 1 to 4 together, on the basis of keeping the gPool and gUnpool layers. Compared with the original graph U-Net, this network's main body fills up the original hollow U-structure part. This architecture takes advantages of the powerful capability of different subsets of network depth and learns the important features of different depths. More importantly, multiple graph U-Net networks within this architecture actually share the same feature extraction encoder part, so it only needs to train all the encoding blocks in the same down-sampling process, and then the features at different levels are restored by different up-sampling paths, which greatly reduces the number of training parameters. It is proved that this architecture is not simply a stack version of graph U-Nets. However, if the original long skip-connection way in graph U-Net is not changed, such an architecture cannot be trained, because except the deepest decoder part, the other shallow decoder parts are disconnected from crossentropy function during the back propagation, and no gradient information will be propagated through this region (the red area in Fig. 3).

The skip-connection in graph U-Net is similar to the residual-connection way, so referring to DenseNet's [40] experience in improving the classification performance of ResNet [41], we introduce the classic dense-connection

Fig. 4 The graph U-Net+ architecture

way. Thus, our graph U-Net+ has both long and short connection ways, which increase the reuse of multi-scale and high-order features, and ensure that the network could be trainable. In addition, we also introduce a deep supervision mechanism, which cannot only improve the performance of the graph U-Net+, but also solve the problem that the gradient cannot propagate back. The specific method is to add a linear layer after the last decoding block of each decoder, and then transmit output directly participate in the calculation of loss function, which is equivalent to supervising the output of the sub graph U-Net of each branch. The deep supervision mechanism integrates the multi-level outputs of graph U-Net+ to obtain better node classification performance. In addition, all the four output parts participate in the training stage, while in the testing stage, because the whole network has been trained well, there is only the forward transmission process, so we can compare which of the four output parts has the best performance and prune the other “branches”. This operation is convenient for corporations to select the appropriate size based on the pre-trained model according to the actual task requirements, thus improving the scalability of our proposed graph U-Net+ in different environments. Figure 4 shows the complete graph U-Net+ architecture with designed skip-connection way and deep supervision mechanism.

3.2 Improved information aggregation layer BGNN

We introduce the GCNII layer and bilinear information aggregator layer, and combine them into the improved information aggregation layer of graph U-Net+'s encoding/decoding blocks.

We improve the GCN layer of graph U-Net to GCNII layer [31], the formula of forward propagation is defined as:

$$X_{\ell+1} = \sigma(((1 - \alpha_{\ell})\hat{P}X_{\ell} + \alpha_{\ell}X_0)((1 - \beta_{\ell})I_N + \beta_{\ell}W_{\ell})) \quad (2)$$

where $\hat{P} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ represents the normalized graph Laplacian matrix; $\hat{A} = A + 2I$ is retained to highlight the features of node itself during information propagation and aggregation process; I_N is the identity matrix which N denotes the number of nodes; W_{ℓ} is the learnable parameter matrix; α_{ℓ} and β_{ℓ} are the hyperparameters that control the initial residual connection and the proportion of identity mapping respectively. $\alpha_{\ell} \in [0, 1]$ represents that each node output by the

GCNII layer contains at least a part of the input feature X_0 , where X_0 refers to the result after dimension reduction through the input linear layer. The principle of setting β_ℓ is to ensure that the attenuation rate of weight matrix increases adaptively as the network stacks more layers. In practice, $\beta_\ell = \log\left(1 + \frac{\lambda}{\ell}\right) \approx \frac{\lambda}{\ell}$, where λ is a hyperparameter. Compared with the GCN layer in graph U-Net, the GCNII layer can alleviate the over-smoothing problem caused by the deepening of the graph network, so as to improve the capability of nodes' representation learning process.

In order to mine and utilize neighbor interactions information between nodes for better representation, the bilinear information aggregator layer [32] is introduced in this paper. Common GCN or GCNII layer can be boiled down to a weighted summation of information by a node over its neighbors, may sometimes loss between nodes of some common properties, and the bilinear information aggregator can capture and strengthen the weak signals by dot product operation, such as the interaction information between neighbors, the common characteristics between friends. The forward propagation formula of bilinear information aggregator is defined as:

$$X_{\ell+1} = \text{BA}(X_\ell, \hat{A}) = \frac{1}{2}(B_\ell)^{-1} \left((\hat{A}X_\ell W_\ell)^2 - \hat{A}(X_\ell W_\ell)^2 \right) \quad (3)$$

where $\hat{A} = A + 2I$; B is the diagonal matrix, $B_{ii} = b_{v_i} = \frac{1}{2}\hat{d}_{v_i}(\hat{d}_{v_i} - 1) = \frac{1}{2}\hat{D}_{ii}(\hat{D}_{ii} - 1)$. b_{v_i} represents the interaction times of node v_i , and the obtained node representation can be normalized to eliminate the deviation caused by node degree. This information aggregation method like bilinear information aggregator layer emphasizes the interaction/contact information between neighbor nodes, i.e. the common features, while dilutes and weakens the difference information.

In this paper, we combine GCNII and bilinear information aggregator together, and construct a more powerful information aggregation layer BGNN:

$$X_{\ell+1} = \text{BGNN}(X_\ell, \hat{A}) = (1 - \gamma) \cdot \text{GCNII}(X_\ell, \hat{A}) + \gamma \cdot \text{BA}(X_\ell, \hat{A}) \quad (4)$$

where the hyperparameter $\gamma \in [0, 1]$ is used to control the aggregation ratio of the outputs produced by two information aggregators mentioned above. When γ is 0, the graph U-Net+ does not consider the interaction information between nodes, and the information aggregation layer degenerates to the single GCNII layer. When γ becomes 1, the information aggregation layer only uses bilinear information aggregators.

3.3 Encoding and decoding blocks

It is well known that the batch normalization is added into CNN based networks during image processing to prevent the over-fitting problem. In our proposed graph U-Net+, following the information aggregation layer BGNN, we introduce the graph node normalization layer NodeNorm [33] into down/up-sampling blocks, which is used to alleviate the over-smoothing problem and improve the node classification performance. NodeNorm layer defines the node-level bias to measure the feature variance of a single node. Taking node v_i as an example, its feature variance at the layer ℓ is:

$$\text{var}_i^\ell = \frac{1}{d_\ell} \sum_{j=1}^{d_\ell} \left(x_{ij}^\ell - \mu_i^\ell \right)^2 \quad (5)$$

where x_{ij}^ℓ represents the j th feature in the representation vector of node v_i ; d_ℓ denotes the dimension size of the representation vector of node v_i ; $\mu_i^\ell = \frac{1}{d_\ell} \sum_{j=1}^{d_\ell} x_{ij}^\ell$ is the mean of all features of node v_i . We hope that in each encoding and decoding block, after information aggregation layer, all nodes can find a suitable representation in a latent embedding space. Through the operation of centralization and then stretching, the new embedding representation of node v_i is defined as:

$$\text{NodeNorm}(x_i^{\ell+1}) = \frac{x_i^\ell}{(\sigma_i^\ell)^{\frac{1}{2}}} \quad (6)$$

where $\sigma_i^\ell = \sqrt{\text{var}_i^\ell}$ is the standard deviation of the node v_i . In the node classification task, the nodes with similar features often belong to the same category, NodeNorm makes the location of node in latent space more close, which means that the distance of similar nodes is smaller, and the classification effect is better.

Fig. 5 The proposed encoding/decoding block of graph U-Net+

Table 1 The statistics of four datasets

	Digg	Twitter	Weibo	OAG
$ V $	279,630	456,626	1,776,950	953,675
$ E $	1,548,126	12,508,413	308,489,739	4,151,463
N	24,428	362,888	779,164	499,848

$|V|$ and $|E|$ represents the number of nodes and edges of each dataset. N indicates the number of target ego-users which are social influence neighborhood

Based on gPool, gUnpool, information aggregation layer BGNN, and graph node normalization layer NodeNorm, we propose our encoding/decoding blocks of graph U-Net+. As shown in Fig. 5, an encoding block contains a gPool layer followed by a BGNN layer and a NodeNorm layer. gPool layer reduce the graph size, while BGNN layer are responsible for aggregating information from each node's neighborhood information and NodeNorm computes a new appropriate representation for nodes in latent embedding space to avoid over-smoothing problem. Each decoding block is composed of a BGNN layer, a NodeNorm layer and a gUnpool layer. The pseudocode Algorithm 1 of entire program running is shown in Appendix, it is an instruction of network training, validating and testing process.

4 Experimental settings

In this section, we give the details of datasets and evaluation experiments. We set up our experiments with large-scale real-world datasets to quantitatively evaluate the proposed graph U-Net+ model. We first introduce these four datasets, then introduce the details of input data. After that, we introduce the evaluation metrics and display the experimental settings.

4.1 Datasets

Our experiments are conducted on the following social media and co-author citation networks: Digg, Twitter, Weibo, and Open-Academic-Graph (OAG). Table 1 lists all statistics of four datasets mentioned above. In this paper, we only consider the undirected relationship (undirected graph) situation.

Digg [42]: Digg¹ is a website that collects articles and stories. It allows users to vote on contents (that is, each story or article) according to personal preference and opinions of their online friends, and the stories that receive more votes will be recommended to the website's home page. This dataset contains history voting data for stories that were promoted to the Digg home page during a month in 2009. For each story, it contains a list of users who have voted on this story up to the data collection time, as well as the UNIX timestamp that each vote occurred, and friend relationships between voters are also retrieved. Digg relationship network has 279,630 nodes and 1,548,126 edges. The social behavior is defined

¹ <https://www.isi.edu/lerman/downloads/digg2009.html>.

Table 2 The sample partitioning of the training/validation/testing sets for the four datasets

Dataset	<i>N</i>	Training	Validation	Testing
Digg	24,428	18,321	3054	3053
Twitter	362,888	272,166	45,361	45,361
Weibo	779,164	584,373	97,396	97,395
OAG	499,848	374,886	62,481	62,481

The sum of three subsets equals *N*

Fig. 6 The node representation consists of three parts: pretrained 64-dim DeepWalk embedding; customized 7-dim social influence metrics, including coreness, eigenvector centrality, clustering coefficient, etc; two hand-crafted features, one indicates the behavior status (0/1), the other indicates whether the user is the ego-user (0/1)

as whether users vote or not. For the entire Digg network, the dataset specifies 24,428 nodes as target ego-users and builds 24,428 subgraphs around them.

Twitter [43]: On July 4, 2012, scientists announced on Twitter that they had discovered a new particle with the features of the elusive Higgs boson. The Stanford team monitored and recorded the news spreading through Twitter² platform. This dataset records the history retweeting of this news among users, including the users' friend relationship network which has 456,626 nodes and 12,508,418 edges. The social behavior is defined as whether the user was involved in retweeting this news. This dataset specifies 362,888 users as target ego-users.

Weibo [44, 45]: Sina Weibo³ is the most popular microblogging site in China right now. The complete dataset collects the activity trail (posting logs) of 1,776,950 users from September 28th, 2012 to October 29, 2012, where social behavior is defined as the user's retweeting actions in Weibo. The entire Weibo network contains 308,489,738 edges. If there is an edge connection between two user nodes, it denotes they have a friend relationship. The dataset choose 779,164 nodes as target ego-users.

OAG: OAG⁴ (Open Academic Graph) dataset is generated by connecting two large academic graph networks: MAG (Microsoft Academic Graph) [46] and AMiner [47], including 20 popular conferences from data mining, information retrieval, machine learning, natural language processing, computer vision, and database research communities.⁵ The social network is defined to be the co-author citation network, in which 953,675 nodes represent scholars, 4,151,463 edges represent collaborative relationships among them. The social behavior is defined to be citation actions—a researcher cites a paper from the above conferences. For the entire OAG network, dataset designates 499,848 nodes as target ego-users. We want to explore how one's citation behaviors are influenced by his or her collaborators.

In this paper, the proportions of the above datasets in training/validation/testing sets are 75%, 12.5% and 12.5% respectively. Table 2 shows the number of target ego-users after specific division.

4.2 Data preparation

Each dataset mentioned above consists of two main parts: adjacency matrix and node representation matrix. The pre-process is described below.

² <https://snap.stanford.edu/data/higgs-twitter.html>.

³ <http://aminer.org/Influencelocality>.

⁴ <https://www.microsoft.com/en-us/research/project/open-academic-graph/>.

⁵ KDD, WWW, ICDM, SDM, WSDM, CIKM, SIGIR, NIPS, ICML, AAAI, IJCAI, ACL, EMNLP, CVPR, ICCV, ECCV, MM, SIGMOD, ICDE, and VLDB.

Adjacency matrix: As shown in Table 1, each dataset has N target ego-users. Because our goal is to predict the future behavior of these target ego-users, so we focus on the scope of target node and its local neighborhood. For one target ego-user, random walk with restart (RWR) algorithm starts from this node, and iteratively travels to its neighborhood with the probability that is proportional to the weight of each edge. Besides, at each step, the walk is assigned a probability 20% to return to the target ego-user or its active neighbor. The RWR runs until it successfully collects 50 nodes from social network, including target ego-user. After sampling, the entire social media network becomes N subgraphs. Each subgraph is defined as an undirected graph type which edge weight is 1.

Node representation matrix: The size of this matrix is $(N \times 50 \times 73)$, each row represents a 73 dimensional feature vector of one node. Figure 6 shows the components of node representation features. The first yellow part is obtained by the network embedding algorithm DeepWalk [19]. The second green part comes from the characteristics of network, and these social influence metrics measures the impact of nodes which objectively reflect the importance of nodes in the network, the structure information of the network and some attributes of nodes themselves. The last blue part contains two hand-crafted features, one represents whether the user is the ego-user (0/1), and the other one represents the behavior status (0/1) of user.

4.3 Evaluation metrics and baselines

Evaluation metrics: To evaluate the performance of our proposed graph U-Net+ model in the social media user behavior prediction task, we utilize four standard metrics which are usually used in the classification task. **Precision** (Prec.) reflects the percentage of the samples that the model actually predicted correctly. **Recall** (Rec.) refers to how many positive instances in samples are predicted correctly, reflecting the success rate of model classification. **F1** (F1-score) is usually used in the binary classification task, it is the weighted harmonic average of precision and recall. **AUC** (Area Under Curve) is also a standard evaluation metric in binary classification task, it refers to the area enclosed by the coordinate axis under the ROC curve, which is plotted with FPR as the horizontal axis and TPR as the vertical axis.

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (9)$$

$$FPR = \frac{FP}{FP + TN}, \quad TPR = \frac{TP}{TP + FN}. \quad (10)$$

Baselines: We compare our proposed graph U-Net+ approach with other five state-of-the-art approaches, including GCN and GAT based DeepInf [17, 23, 25], PPNP and APPNP [26, 27], and graph U-Net [30]. The results of GCN and GAT based DeepInf, PPNP and APPNP are directly cited from existing papers [25, 26]. We further reproduce and test the graph U-Net model on all datasets, and the results are saved for performance comparison.

4.4 Hyper-parameter settings and implementation details

We build a graph U-Net+ framework which contains all modules mentioned in Sect. 3 with a depth of 4, and test its performance on four datasets. Each node has an initial feature dimension length of 73, so we add a linear layer with the size of 73×64 as input layer (grey block). The size of parameter matrix used for the feature dimension transformation of information aggregation layer in each downsampling encoder module is 64×64 . Correspondingly, in the upsampling decoder module, the size of parameter matrix is set to be 64×64 , since the skip connection adopts the summation way. Before the output is sent into the crossentropy loss function, we set another linear layer as the output layer. Due to the task is a binary classification problem, so the size of this linear layer is set to be 64×2 (grey block). For detailed experiment configuration, we adopt Mish [48] as nonlinearity activation function. All the parameters are initialized with

Table 3 Social behavior prediction performance of different methods on four large real-world datasets

Dataset	Methods	AUC	Prec.	Rec.	F1
Digg	DeepInf-GCN	84.15	58.76	67.61	62.88
	PPNP-0.9	89.03	68.12	71.51	69.77
	APPNP-0.9	89.06	68.32	70.43	69.36
	DeepInf-GAT	90.65	66.82	78.49	72.19
	Graph U-Net	90.59	71.84	76.21	73.96
	Graph U-Net+	94.70	76.69	78.47	77.57
Twitter	DeepInf-GCN	76.60	44.31	66.74	53.26
	PPNP-0.9	77.62	46.94	65.58	54.72
	APPNP-0.9	77.71	47.22	65.49	54.87
	DeepInf-GAT	80.22	48.41	69.08	56.93
	Graph U-Net	79.10	48.76	70.39	57.61
	Graph U-Net+	83.43	51.48	73.10	60.41
OAG	DeepInf-GCN	63.55	30.23	74.36	43.03
	PPNP-0.9	65.89	33.58	64.57	44.18
	APPNP-0.9	65.96	33.35	66.53	44.21
	DeepInf-GAT	71.79	40.77	60.97	48.86
	Graph U-Net	66.48	33.19	69.53	44.93
	Graph U-Net+	73.44	39.83	63.99	49.10
Weibo	DeepInf-GCN	76.85	42.44	71.30	53.21
	PPNP-0.9	–	–	–	–
	APPNP-0.9	–	–	–	–
	DeepInf-GAT	82.72	48.53	76.09	59.27
	Graph U-Net	81.29	46.01	76.24	57.38
	Graph U-Net+	84.58	49.81	79.51	61.25

The bold values represent the best performance in that column of metrics. It is obviously that our model's performance is better than other approaches in most situations

Both graph U-Net and graph U-Net+ have a depth of 4. '–' indicates that the experimental results cited are missing in paper [26] (%)

Glorot initialization [49] and trained by Adam optimizer [50] with learning rate 0.01, weight decay $5e^{-4}$, dropout rate 0.2, α 0.2 and λ 0.5 for GCNII layer, pooling ratio 0.5 for graph pooling layer, γ 0.3 for each information aggregation layer of encoding and decoding blocks. We set random seed to be 42. After the data was shuffled, we divide all subgraphs in each dataset into training, validation and test at the proportions of 75%, 12.5%, 12.5%, respectively. We allow our model to run at most 1000 epochs over the training data, and the best model was selected by early stopping strategy with 100 patience by comparing loss on the validation sets. The mini-batch size is set to be 1024 across all datasets.

All designed experiments are based on the Pytorch1.6.0 under Linux Ubuntu16.04 system environment. The hardware configurations of the experimental platform are Intel® Xeon® E5-2680 v4 @ 2.40GHz CPU, and Nvidia GTX2080Ti GPU with 11019MiB memory. We also implement a graph based deep learning library called Pytorch-Geometric which helps improving the performance and reducing time.

5 Results and analysis

In this section, we compare our proposed graph U-Net+ with previous state-of-the-art models on social behavior prediction (node classification) task. Experimental results show that our method achieve promising results on four common large real-world datasets and has good capability to overcome the over-smoothing problem. Some ablation studies are performed to examine the contributions of the improvements proposed in Sect. 3.

Table 4 Keep the original network layers of graph U-Net unchanged, and only change the network architecture from graph U-Net to graph U-Net+ (%)

Dataset	Network architecture	AUC	Prec.	Rec.	F1
Digg	Graph U-Net	90.59	71.84	76.21	73.96
	Graph U-Net+	91.33	73.21	76.84	74.98
Twitter	Graph U-Net	79.10	48.76	70.39	57.61
	Graph U-Net+	80.23	49.64	71.81	58.70
OAG	Graph U-Net	66.48	33.19	69.53	44.93
	Graph U-Net+	67.55	34.23	69.12	45.79
Weibo	Graph U-Net	81.29	46.01	76.24	57.38
	Graph U-Net+	82.10	46.96	76.67	58.25

Table 5 Based on the experiment in Table 4, we only change the GCN layer of graph U-Net+ into GCNII layer (%)

Dataset	Graph U-Net+ with GCN/ GCNII	AUC	Prec.	Rec.	F1
Digg	With GCN	91.33	73.21	76.84	74.98
	With GCNII	92.80	74.44	77.60	76.49
Twitter	With GCN	80.23	49.64	71.81	58.70
	With GCNII	81.68	51.57	73.05	60.46
OAG	With GCN	67.55	34.23	69.12	45.79
	With GCNII	69.13	35.86	70.24	47.48
Weibo	With GCN	82.10	46.96	76.67	58.25
	With GCNII	83.77	47.80	77.99	59.27

5.1 Performance study

We conduct a verification experiment on model validity, and present the AUC, precision, recall, and F1 results of graph U-Net+ with the other compared approaches, such as GCN-based and GAT-based DeepInf, PPNP and APPNP, and graph U-Net, on four large real-world datasets in Table 3. Both graph U-Net and graph U-Net+ have a depth of 4. We can observe from the results that our proposed graph U-Net+ achieves better performance than other networks. Especially on two key metrics AUC and F1, the graph U-Net+ outperforms than other methods consistently. In this experiment, except for the result of graph U-Net which is reconstructed by us, other results are directly cited from these two paper [25, 26]. Our proposed model is composed of GCNII and bilinear information aggregator module without involving more advanced graph convolution layers like GAT, but AUC and F1 metrics on four datasets show that graph U-Net+ performs better than graph self-attention based DeepInf-GAT. This is because the encoder-decoder architecture has better performance in pixel-level or node-level classification. Similarly, compared to the GCN-based methods DeepInf-GCN, APPNP and graph U-Net, our graph U-Net+ not only has the advanced network architecture, but also because we improve the graph convolution approach in encoding and decoding blocks. The additional information aggregation mechanism is introduced, so that the fusion process of node information is more reasonable and robust. In the other hand, the graph normalization technique can also greatly inhibit the occurrence of over-smoothing problems. As for the less performance of precision and recall metrics on OAG dataset, we believe that the reason why graph U-Net+ model proposed in this paper performs poorly on OAG dataset may be that the other three datasets are all standard social networks based on the real world media, in which user interaction and friend relationship are more close to the research interest of this paper. The OAG dataset is a “social network”-like dataset extracted from the citation and co-author networks, their network structures are totally different, so the precision and recall metrics did not outperform other methods, but the more convincing AUC and F1 metric proved the effectiveness of our method. Also in the graph node classification tasks, researchers mainly focus on the AUC and F1 metrics. F1 takes into account both the precision and recall rates of the classification model, which can be regarded as the weighted sum of the precision and recall rates of the model. Therefore, higher F1 values represent better model performance. Specially, the unique architecture design and the use of deep supervision mechanism leads to marked improvement over graph U-Net. When compared to graph U-Net directly through AUC, our graph U-Net+ significantly improves

Table 6 Social behavior prediction performance of graph U-Net+ with or without node bilinear information aggregator on AUC and F1 metrics (%)

Dataset	Graph U-Net+ w/wo bilinear aggregator	AUC	F1
Digg	Without bilinear aggregator	94.10	76.95
	With bilinear aggregator	94.70	77.57
Twitter	Without bilinear aggregator	82.96	60.01
	With bilinear aggregator	83.43	60.41
OAG	Without bilinear aggregator	72.77	48.50
	With bilinear aggregator	73.44	49.10
Weibo	Without bilinear aggregator	83.92	59.77
	With bilinear aggregator	84.58	61.25

Table 7 Social behavior prediction performance of graph U-Net+ with or without NodeNorm layer on AUC and F1 metrics (%)

Dataset	Graph U-Net+ w/wo NodeNorm	AUC	F1
Digg	Without NodeNorm	92.71	75.60
	With NodeNorm	94.70	77.57
Twitter	Without NodeNorm	81.69	58.70
	With NodeNorm	83.43	60.41
OAG	Without NodeNorm	71.49	47.26
	With NodeNorm	73.44	49.10
Weibo	Without NodeNorm	82.44	59.38
	With NodeNorm	84.58	61.25

Table 8 Social behavior prediction performance of graph U-Net+ with or without deep supervision mechanism on AUC and F1 metrics (%)

Dataset	Graph U-Net+ w/wo deep supervision	AUC	F1
Digg	Without deep supervision	93.84	77.06
	With deep supervision	94.70	77.57
Twitter	Without deep supervision	82.71	59.95
	With deep supervision	83.43	60.41
OAG	Without deep supervision	72.66	48.87
	With deep supervision	73.44	49.10
Weibo	Without deep supervision	83.70	60.61
	With deep supervision	84.58	61.25

The bold values represent the best performance in that column of metrics. It is obviously that our model's performance is better than other approaches in most situations

performance on all four datasets by margin of 4.11%, 4.33%, 6.96%, and 3.29%, respectively. These results demonstrate the effectiveness of graph U-Net+ in node classification and social behavior prediction task.

5.2 Ablation study

In this section, we investigate the contributions of five improvements to the performance of graph U-Net+. In order to study the effectiveness of proposed network architecture, we conduct experiment by keeping all GCN, gPool, and gUnpool layers within graph U-Net+. Note that the only difference between our graph U-Net+ and graph U-Net is the structure of networks. Table 4 shows the results that illustrate graph U-Net+ have better effectiveness than graph U-Net, our designed architecture is a better version of encoder–decoder architecture. When considering the difference between the two models in terms of architecture, graph U-Net+ enable higher level feature encoding thereby resulting in better performance due to the use of deep-supervision mechanism and dense-like skip connection way.

On the basis of experiment conducted in Tables 4, 5 provides the comparison results between graph U-Net+ with GCN or GCNII layers. We keep the gPool and gUnpool layers unchanged, and only change the GCN layer to GCNII layer. The results show that graph U-Net+ architecture with GCNII has better performance over graph U-Net+ with GCN by

margins of average 1.54% on AUC and 1.50% on F1 across all datasets. Note that the graph U-Net+ architecture in Table 5 does not contain bilinear information aggregator module and NodeNorm layer.

The following two experiments of Tables 6 and 7 are designed to verify the functional validity of the bilinear information aggregator and NodeNorm layer we introduced in graph U-Net+. The results show that these two layers with GCNII module are helpful to improve the prediction performance and reduce the effect of over-smoothing problem.

In the last experiment, we try to verify the effectiveness of the deep supervision mechanism added to graph U-Net+ by removing other “branch” output connecting to the loss function. We only keep output of the deepest decoder, because it carries much more information contributed by encoding blocks through skip-connection way. The version of “without Deep Supervision” represents that graph U-Net+ only contains the output after linear layer of encoder with longest path. The results in Table 8 provide the evidence that graph U-Net+ with deep supervision mechanism performs better by margins of average 0.81% on AUC and 0.46% on F1 across all datasets. All experimental results yields significantly performance improvement.

6 Conclusion

In this work, we study the social media user behavior prediction problem. This paper explores and enriches more possibilities of behavior prediction model, and put forward graph U-Net+ model with more complex structure but better performance on the basis of graph U-Net. We retain the both gPool and gUnpool operations from graph U-Net. Our graph U-Net+ architecture can be thought of a stacked version of graph U-Net at different depths, and change its skip connection's way from residual to dense. These two changes help model absorb more features while restoring and retaining as much of the original and high-level information as possible. In order to make the model trainable, we also add a deep supervision mechanism to the final loss function. Furthermore, we expand and enrich the internal structure of encoder and decoder. The original graph U-Net only contains a single graph convolution in both encoder and decoder blocks, but in graph U-Net+, we replace it with a combination of GCNII and bilinear information aggregator modules to enrich the learning capability of node representation. The graph normalization technique has also been introduced into our encoder and decoder to help our model overcomes the over-smoothing problem. We test our model on four social media and co-author citation networks: Digg, Twitter, Weibo and OAG. Experimental results demonstrate that our proposed graph U-Net+ has achieved encouraging performance and significantly outperforms previous methods.

Acknowledgements This work was supported in part by National Natural Science Foundation of China under Grants 61771322, 61871186 and 61375015, and in part by the Fundamental Research Foundation of ShenZhen under Grant JCYJ20190808160815125.

Authors' contributions ZY designed the network. ZY and WC performed the extensive experiments. ZY, WC and JJ analysed the data and results, and wrote the manuscript equally. All authors read and approved the final manuscript.

Data availability statement All data generated or preprocessed during this study are included in this published article [25]. The four datasets have been deposited in baidu pan https://pan.baidu.com/share/init?sur=YX3cHYaK_7UuX4qEnqgo9w with the access code 242g. If you are interested in un-preprocessed data, please download them from the following links. <https://www.isi.edu/lerman/downloads/digg2009.html> for Digg, <https://snap.stanford.edu/data/higgs-twitter.html> for Twitter, <https://www.aminer.cn/influencelocality> for Weibo, and <https://www.microsoft.com/en-us/research/project/open-academic-graph/> for OAG.

Declarations

Ethics approval and consent to participate This manuscript follows the principles and guidelines of the Committee on Publication Ethics (COPE), and did not be submitted to more than one journal for simultaneous consideration.

Competing interests The authors declare that they have no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix: The pseudocode of entire program running

Algorithm 1: An instruction of program codes including training, validation, and testing stages

Input: $A_{N \times 50 \times 50}$: subgraph adjacency matrix; $X_{N \times 50 \times 73}$: subgraph feature matrix; $L_{N \times 1}$: label matrix of behavior status as 0 or 1;

Output: Optimal model parameters θ_{opt} ;

Initialization: batch size: $BS=1024$; training set: $D_{tr}=N \times 0.75$; validation set: $D_{val}=N \times 0.125$; testing set: $D_{te}=N \times 0.125$; epoch: $E=1000$; early stopping patience: $P=100$; initialize model parameters θ with Glorot initialization, and shuffle(X, A) with seed 42;

```

i=1;                                     /* i denotes the index of epoch */
p=1;
lossmin=0;
while i ≤ E do                               /* epochs loop */
    lossbatch=0;
    for j=1; j ≤ Dtr/BS; j++ do           /* training batches loop */
        compute output of decoder#1: O1=Graph U-Net+(Xtr, Atr);
        compute output of decoder#2: O2=Graph U-Net+(Xtr, Atr);
        compute output of decoder#3: O3=Graph U-Net+(Xtr, Atr);
        compute output of decoder#4: O4=Graph U-Net+(Xtr, Atr);
        lossbatch+=LossFunction(O1, O2, O3, O4, Ltr);
    end
    losstr=lossbatch/Dtr;                 /* training loss of epoch i */
    update network parameters θ through back propagation;
    i+=1;
    lossbatch=0;
    for k=1; k ≤ Dval/BS; k++ do         /* validation batches loop */
        compute output of decoder#1: O1=Graph U-Net+(Xval, Aval);
        compute output of decoder#2: O2=Graph U-Net+(Xval, Aval);
        compute output of decoder#3: O3=Graph U-Net+(Xval, Aval);
        compute output of decoder#4: O4=Graph U-Net+(Xval, Aval);
        lossbatch+=LossFunction(O1, O2, O3, O4, Lval);
    end
    lossval=lossbatch/Dval;             /* validation loss of epoch i+1 */
    i+=1;
    lossbatch=0;
    while i==3 do
        lossmin=lossval;                /* assign first lossval to lossmin */
    end
    if lossval ≥ lossmin && p ≤ P then
        p+=1;
        Continue;
    else if lossval ≥ lossmin && p>P then    /* complete Early Stopping condition */
        break;
    else
        for l=1; l ≤ Dte/BS; l++ do       /* testing batches loop */
            compute output of decoder#1: O1=Graph U-Net+(Xte, Ate);
            compute output of decoder#2: O2=Graph U-Net+(Xte, Ate);
            compute output of decoder#3: O3=Graph U-Net+(Xte, Ate);
            compute output of decoder#4: O4=Graph U-Net+(Xte, Ate);
            lossbatch+=LossFunction(O1, O2, O3, O4, Lte);
        end
        losste=lossbatch/Dte;           /* testing loss of epoch i+2 */
        update lossmin as lossval;
        p=1;
        i+=1;
        save and update the optimal model parameters θopt;
        Continue;
    end
end
Return Optimal model parameters θopt;

```

References

1. Sun J, Tang J. A survey of models and algorithms for social influence analysis. In: *Social network data analytics*. Boston: Springer; 2011. p. 177–214.
2. Tang J, Sun J, Wang C, Yang Z. Social influence analysis in large-scale networks. In: *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*; 2009. p. 807–16.
3. Li K, Zhang L, Huang H. Social influence analysis: models, methods, and evaluation. *Engineering*. 2018;4(1):40–6.
4. Bakshy E, Eckles D, Yan R, Rosenn I. Social influence in social advertising: evidence from field experiments. In: *Proceedings of the 13th ACM conference on electronic commerce*; 2012. p. 146–61.
5. Goeldi A. Website network and advertisement analysis using analytic measurement of online social media content. US Patent 7,974,983; 2011.
6. Pálovics R, Benczúr AA, Kocsis L, Kiss T, Frigó E. Exploiting temporal influence in online recommendation. In: *Proceedings of the 8th ACM conference on recommender systems*; 2014. p. 273–80.
7. Kim YA, Srivastava J. Impact of social influence in e-commerce decision making. In: *Proceedings of the ninth international conference on electronic commerce*; 2007. p. 293–302.
8. He J, Chu WW. A social network-based recommender system (SNRS). In: *Data mining for social network data*. Boston: Springer; 2010. p. 47–74.
9. Richardson M, Domingos P. Mining knowledge-sharing sites for viral marketing. In: *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining*; 2002. p. 61–70.
10. Leskovec J, Adamic LA, Huberman BA. The dynamics of viral marketing. *ACM Trans Web (TWEB)*. 2007;1:5-es.
11. Domingos P. Mining social networks for viral marketing. *IEEE Intell Syst*. 2005;20(1):80–2.
12. He Z, Cai Z, Wang X. Modeling propagation dynamics and developing optimized countermeasures for rumor spreading in online social networks. In: *2015 IEEE 35th international conference on distributed computing systems*. IEEE; 2015. p. 205–14.
13. Bond RM, Fariss CJ, Jones JJ, Kramer ADI, Marlow C, Settle JE, Fowler JH. A 61-million-person experiment in social influence and political mobilization. *Nature*. 2012;489(7415):295–8.
14. Matsubara Y, Sakurai Y, Prakash BA, Li L, Faloutsos C. Rise and fall patterns of information diffusion: model and implications. In: *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining*; 2012. p. 6–14.
15. Jie T, Wu S, Sun J. Confluence: conformity influence in large social networks. In: *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining*; 2013. p. 347–55.
16. Li C, Ma J, Guo X, Mei Q. Deepcas: an end-to-end predictor of information cascades. In: *Proceedings of the 26th international conference on World Wide Web*; 2017. p. 577–86.
17. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907); 2016.
18. Hamilton WL, Ying R, Leskovec J. Inductive representation learning on large graphs. arXiv preprint [arXiv:1706.02216](https://arxiv.org/abs/1706.02216); 2017.
19. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*; 2014. p. 701–10.
20. Grover A, Leskovec J. node2vec: scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*; 2016. p. 855–64.
21. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. Line: large-scale information network embedding. In: *Proceedings of the 24th international conference on world wide web*; 2015. p. 1067–77.
22. Qiu J, Dong Y, Ma H, Li J, Wang K, Tang J. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In: *Proceedings of the eleventh ACM international conference on web search and data mining*; 2018. p. 459–67.
23. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. arXiv preprint [arXiv:1710.10903](https://arxiv.org/abs/1710.10903); 2017.
24. Thekumparampil KK, Wang C, Oh S, Li LJ. Attention-based graph neural network for semi-supervised learning. arXiv preprint [arXiv:1803.03735](https://arxiv.org/abs/1803.03735); 2018.
25. Qiu J, Tang J, Ma H, Dong Y, Wang K, Tang J. Deepinf: social influence prediction with deep learning. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*; 2018. p. 2110–9.
26. Leung CK, Cuzzocrea A, Mai JJ, Deng D, Jiang F. Personalized deepinf: enhanced social influence prediction with deep learning and transfer learning. In: *2019 IEEE international conference on big data (Big Data)*. IEEE; 2019. p. 2871–80.
27. Klicpera J, Bojchevski A, Günnemann S. Predict then propagate: graph neural networks meet personalized pagerank. arXiv preprint [arXiv:1810.05997](https://arxiv.org/abs/1810.05997); 2018.
28. Ma X, Wu J, Xue S, Yang J, Sheng QZ, Xiong H. A comprehensive survey on graph anomaly detection with deep learning. arXiv preprint [arXiv:2106.07178](https://arxiv.org/abs/2106.07178); 2021.
29. Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. In: *International conference on medical image computing and computer-assisted intervention*. Springer; 2015. p. 234–41.
30. Gao H, Ji S. Graph u-nets. In: *International conference on machine learning*. PMLR; 2019. p. 2083–92.
31. Chen M, Wei Z, Huang Z, Ding B, Li Y. Simple and deep graph convolutional networks. In: *International conference on machine learning*. PMLR; 2020. p. 1725–35.
32. Zhu H, Feng F, He X, Wang X, Li Y, Zheng K, Zhang Y. Bilinear graph neural network with neighbor interactions. In: *Proceedings of the twenty-ninth international joint conference on artificial intelligence (IJCAI)*, vol. 5; 2020.
33. Zhou K, Dong Y, Wang K, Lee WS, Hooi B, Xu H, Feng J. Understanding and resolving performance degradation in graph convolutional networks. arXiv preprint [arXiv:2006.07107](https://arxiv.org/abs/2006.07107); 2020.
34. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for quantum chemistry. In: *International conference on machine learning*. PMLR; 2017. p. 1263–72.
35. Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. arXiv preprint [arXiv:1606.09375](https://arxiv.org/abs/1606.09375); 2016.

36. Simonovsky M, Komodakis N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 3693–702.
37. Ying R, You J, Morris C, Ren X, Hamilton WL, Leskovec J. Hierarchical graph representation learning with differentiable pooling. arXiv preprint [arXiv:1806.08804](https://arxiv.org/abs/1806.08804); 2018.
38. Su X, Xue S, Liu F, Wu J, Yang J, Zhou C, Hu W, Paris C, Nepal S, Jin D, et al. A comprehensive survey on community detection with deep learning. arXiv preprint [arXiv:2105.12584](https://arxiv.org/abs/2105.12584); 2021.
39. Liu F, Xue S, Wu J, Zhou C, Hu W, Paris C, Nepal S, Yang J, Yu PS. Deep learning for community detection: progress, challenges and opportunities. arXiv preprint [arXiv:2005.08225](https://arxiv.org/abs/2005.08225); 2020.
40. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 4700–8.
41. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 770–8.
42. Hogg T, Lerman K. Social dynamics of digg. EPJ Data Sci. 2012;1(1):1–26.
43. De Domenico M, Lima A, Mougél P, Musolesi M. The anatomy of a scientific rumor. Sci Rep. 2013;3(1):1–9.
44. Zhang J, Liu B, Tang J, Chen T, Li J. Social influence locality for modeling retweeting behaviors. IJCAI. 2013;13:2761–7.
45. Zhang Jing, Tang Jie, Li Juanzi, Liu Yang, Xing Chunxiao. Who influenced you? Predicting retweet via social influence locality. ACM Trans Knowl Discov Data (TKDD). 2015;9(3):1–26.
46. Dong Y, Ma H, Shen Z, Wang K. A century of science: globalization of scientific collaborations, citations, and innovations. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining; 2017. p. 1437–46.
47. Jie T, Jing Z, Yao L, Li J, Zhang L, Su Z. Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining; 2008. p. 990–8.
48. Misra MD. A self regularized non-monotonic activation function. arXiv preprint [arXiv:1908.08681](https://arxiv.org/abs/1908.08681); 2019.
49. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR workshop and conference proceedings; 2010. p. 249–56.
50. Kingma DP, Adam JB. A method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980); 2014.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.