Progress in Artificial Intelligence (2022) 11:411–423
https://doi.org/10.1007/s13748-022-00293-3

REGULAR PAPER

Improving graph prototypical network using active learning

Mona Solgi[1] · Vahid Seydi[2]

Received: 27 December 2021 / Accepted: 17 September 2022 / Published online: 10 October 2022

Abstract
Due to the growth of using various devices and applications in modern life, the amount of data available is skyrocketing, but labeling all of this data is beyond the reach of data scientists. Thus, it is necessary to categorize data with a small amount of labeled data. In fact, it should be possible to prioritize data for labeling. To achieve this goal in this study, we have used few-shot learning with active learning and also used the power of graph convolutional networks in classifying data with a graphical structure. To implement the proposed model, we use two graph convolutional networks in parallel to calculate the embedding and the importance of each node. Using the output of both networks, we create prototypes of classes, and then, we classify them according to the distance of each node of these prototypes. We have also used active learning to select data more intelligently, which improves the overall model performance. As well as this, we have tested our proposed model in the field of electronic commerce for tagging goods in big online stores, which encounter a large number of diverse products, where high accuracy categorization in a short time without the interference of human factor and with the help of artificial intelligence is needed to reduce costs. The results of implementing the model on the Amazon dataset and its comparison with the state-of-the-art models in this field show the superiority of our method.

Keywords Data classification · Few-shot learning · Active learning · Graph convolutional network · Product tagging · Online shopping

1 Introduction

There are many methods for data classification, among which using machine power and artificial intelligence to determine the category of each data can be mentioned. Especially when the data volume is very large, using a method with high accuracy and spending less time cost will be very significant.

Data classification is one of the hot topics in the field of machine learning. Systems need data with specific categories for learning, so-called labeled data, through which they can classify unlabeled data. Gathering labeled data in any field is very time-consuming and costly, and data are growing in every category. A way in which new data can be assigned to a particular category with a small amount of labeled data is notable.

One of these methods is to classify data using learning from a small number of samples. The purpose of these algorithms is to train a classifier so that it can classify samples that have not been seen before using only a limited number of labeled training samples (selected from the target data set) [1]. In this method, with the addition of a new category, it is not required to collect thousands of labeled samples and retrain the network.

The key to deal with unfamiliar and new categories is to transfer knowledge gained from familiar data to unknown data. One of the patterns of knowledge transfer, which helps network learning to cluster data, is using implicit knowledge representation such as Semantic Embedding, in which a vector representation of different categories is learned using textual data, and then, a mapping between the vector representation and the data classifier is learned. Graph convolutional network (GCN) method can be used to perform the above classification method. GCN is a powerful type of neural network designed for direct work on graphs, which strengthens the information structure of neural networks.

Active learning method can be used to improve the performance of the proposed learning networks with a small

✉ Vahid Seydi
V.seydi@bangor.ac.uk

1 Department of Information Technology Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

2 Centre for Applied Marine Sciences, School of Ocean Sciences, Bangor University, Menai Bridge, UK

number of examples. In this type of learning, the system can request the label of one of the unlabeled data. If trained, the network will recognize and request the sample label that is most useful for forecasting, thereby reducing the cost function.

One of the applications of data classification is in the e-commerce field. This market has found a special status and importance due to the growth of technologies and the availability of platforms. The impacts that influencers and social networks have gained today also have led to a change in people's buying patterns which increase sales in online markets [2]. On the other hand, in order to be able to stay in the field and be successful, these markets need advanced strategies, one of which is to have a suitable strategy and model for the appropriate classification and product tagging, which is an important and ongoing issue in online marketing. Proper product classification plays a major role in searching and comparing products offered in electronic markets and will also lead to a visual shopping experience, giving the user the ability to find the product, and most importantly increase sales [3]. Especially in the case of large online stores that offer a wide range of products, properly classified data are an important asset and a competitive advantage.

In this article, we have used GCNs to learn embedding and calculate the importance of network nodes, and we have used the output obtained in the tasks created in learning with a small number of instances to create prototypes of classes, and in addition, we have used active learning to intelligently select samples used in few-shot tasks to select more valuable samples to create more accurate prototypes of classes and thus better classification.

In the end, we have compared the results obtained from the model with the proposed models that have been implemented on the Amazon electronic dataset, the results show a significant increase in the experiment's accuracy of the proposed model and its superiority over other models.

2 Related works

2.1 Graph convolutional networks (GCNs)

Graph neural networks are in fact a natural generalization of convolutional networks to nonEuclidean diagrams. GCNs were first proposed in 2016 [4] by Thomas Kipf and Max Welling, inspired by semi-supervised learning on graph-structured data as well as neural networks applied to graphs. The proposed method in the given article was based on spectral graph convolution neural network. These networks form by putting several layers of convolution graphs together. After aggregating features in each layer, a non-linear function is applied to it. In 2018 and 2019, GCN has been developed by other people in various articles in terms of efficiency, analysis and simplification.

In [5–8], the recursive learning problems in GCN and the consequent need for high memory usage and impracticality for large graphs have been explored. Some ideas were put forward that could be implemented for deeper networks. In fact, in these models, the number of neighboring nodes is limited, which is better in terms of the need for computational resources and memory usage than other methods that use the whole graph. In 2018, [9], several instances of GCNs on pairs of nodes discovered at different distances through Random Walks were taught, and their outputs were combined, optimizing the target subject classification.

In 2018, DGCNs were mentioned in [10]. In this paper, two GCN networks were developed in parallel to embed knowledge in local and global compatibility, in which the parameters were shared between the two networks. The authors of [11] in 2018 evaluated the low performance of GCN for large-scale graphs and used the batch algorithm in GCN to solve this problem.

In 2018, limitations of GCNs when there are few labeled data were analyzed in [12]. The paper proposed combining Co-Training using Random Walk and Self Training in GCN, to identify reliable nodes, indicating the closest neighbors to the labeled node of each class. The parameters are also optimized in the training phase so that no additional labeled data are required for validation.

In 2019, the authors of [13] analyzed the GCN and theoretically examined the stability of the network and guaranteed its generality. In this paper, the experiments showed that the stability of the algorithm depends on the largest specific value of the graph convolution filter.

In 2019, the authors of [14] addressed the issue of GCN containing unnecessary complexity and additional computations. In SGCN, the feature collection process has become a simple linear propagation. Besides, the number of learnable parameters (filters) has also decreased. Experiments on citation network datasets show that this method performed equally well and, in some cases, slightly better than GCN.

2.2 (FSL) Few-shot learning

Many researchers have generalized deep learning approaches to solve FSL problems. Most of these approaches used a meta-learning or learning-to-learn strategy, which means that they extract transferable knowledge from the previous task or some auxiliary tasks, such as the transfer learning method [15].

Few-shot learning can be divided into two general categories of meta-learning and metric learning. In the meta-learning method, an algorithm is taught by several learning tasks. Each task contains a set of supports that mimics an N-way, K-shot classification problem. Along with the support set, there is a query set that includes samples of unseen tasks, which are used to test the network's accuracy. Model parameters are updated in each step, based on a randomly selected training task. The loss function of the query set measures the performance of the training task. Since the network presents a different task at each step of the process, it must learn how to distinguish data classes in general rather than a specific set [1]. The agnostic model (MAML) [16] and the LSTM-based few-shot optimization learning method [17] are in this category. The major problem with the proposed methods is that these approaches require fine-tuning to the target issues.

Another common method of FSL is the metric learning method. Metric learning-based methods solve few-shot classification problems by" learning to compare." Algorithms seek to learn embedding in which the data vector is not primarily affected by changes within the class, but preserves class information. Preliminary studies focused on binary comparators that take two samples in parallel and determine if they belong to the same or different classes [1]. Matching networks [18], prototypical networks [19], relation networks [20] and Siamese networks [21] belong to the metric learning approach. Matching networks no longer need to be fine-tuned to adapt to new class types, but the problem with this method is that the algorithm is not flexible in data heterogeneity, indicating that if there are more instances for a class, it prefers that class. In prototype networks, averaging in class samples flexes the method against the problem of data heterogeneity [15].

2.3 Active learning

Active learning is a technique in which the learning algorithm participates in selecting its training data and tries to limit the amount of labeled data by allowing the algorithm to select its training samples [22]. There are several ways to select data in this type of learning, such as the Membership Query Synthesis method, which requests label for any unlabeled sample. These unlabeled items can be produced by the learner itself [23]. In the stream-based selective sampling method introduced in [24], assuming that obtaining an unlabeled sample is charge-free, after obtaining such a sample, the model decides whether this sample should be labeled by the oracle or not. Another approach is to find the uncertainty zone [25], meaning that if both models agree on all labeled data, but disagree in some unlabeled cases, that sample is in the uncertainty zone. The next method is pool-based sampling, which is suggested by [26]. A large collection of unlabeled items can be collected, and then, the queries are selectively taken from the unlabeled item collection [27]. Therefore, the model decides which items have the most information in the pool; thus, it should be labeled by an oracle.

3 The proposed method

3.1 Relationship with existing models

We have developed our active learning model upon graph prototypical network (GPN) [28] which is based on graph convolutional network (GCN) [4] and also used the ideas that were previously proposed in several papers including prototypical network [19] and relation network [20].

GPN proposed a graph meta-learning framework to solve the problem of few-shot learning in node classification on attributed networks. It learns a transferable learning method in which labels of nodes will be predicted according to the distance to a class prototype. In other words, the less distance to a class, the more similarity the node has to that class. This framework consists of two pivotal networks, that exploit GCNs and work together seamlessly. The first one is named Encoder Network to compress the data in the network and extract the feature embedding of nodes. The second one estimates the importance of labeled nodes and maps each node to a scalar score parallelly. By doing so, the output of two networks yields features of the labeled node along with their scores that are used to create prototypes of every class. It performs meta-learning on a semi-supervised pool and extracts meta-knowledge gradually from an attributed network, which is in the form of a graph, to generalize learning ability more effectively on few-shot classification tasks.

As mentioned above, the essential part of GPN is based on graph neural networks (GCNs). Convolution neural network generalizes convolution operation on spectral-domain to learn network representation and then was used on graphs. Graph neural network learns an aggregator function to aggregate features from neighboring nodes instead of training embedding for each node since it is assumed that connected nodes have similar features and consequently the same label. It represents an approach for semi-supervised learning on data in a graph-structured to classify nodes. To improve node classification, a gathering of node features is done in every layer. The propagation rule is defined as below:

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

Here, $H^{l} \in \mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ is the matrix of activation in the lth layer; $H^{0} = X$, X is a matrix of node feature vectors, σ is an activation function, $\hat{A}$ is $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ which $\tilde{A}$ is $\tilde{A} = A + I_N$ that is the adjacency matrix with a self-loop added for nodes to have features of themselves too and $\tilde{D} = \sum_j \tilde{A}_{ij}$ to prevent exploding vanishing gradients in deep neural networks.

A two-layers GCN is described as:

$$Z = f(X, A) = Softmax\left(\hat{A}ReLU\left(\hat{A}XW^{(0)}\right)W^{(1)}\right) \tag{2}$$

Cross-entropy error is then evaluated on labeled samples to minimize the loss and $W^{(0)}$ and $W^{(1)}$ are learned by using gradient descent through the full batch in every training iteration.

The idea of using two modules in parallel is according to relation network [20]. In this work, two modules were used to tackle the problem of a few-shot setting by learning a distance metric during meta-learning. The first one is to obtain node embedding and another one for estimating relation scores within the episode. Firstly, samples in the support set and query set are fed through a four-blocks convolution network ($f\varphi$) to produce feature maps. Secondly, after concatenation of samples (x_i) and queries (xj), a two-convolution layer network yields a scalar between (0,1), which is called relation score (r), to show the similarity between x_i and xj.

$$r_{ij} = g_{\emptyset}\left(C\left(f_{\varphi}(x_i)\right), \left(f_{\varphi}(x_j)\right)\right) \tag{3}$$

Here, $g_{\emptyset}$ is the relation module and C (0,0) is the concatenation of input features. This yields an N relation that is the number of classes we have in an episode, which means that the framework is designed to learn by comparing node embeddings between query nodes and those in support sets (that each belongs to a specific class) to classify queries according to the highest relation score to support samples.

In GPN, the prototypes of classes were created according to prototypical networks which were proposed to deal with over-fitting issues in few-shot learning. This metric-based approach trains episodically, and each one selects support and query samples from training classes randomly. It learns a non-linear mapping of node embeddings within a neural network to create class prototypes by computing the mean of support embedding for each class. After that, embedded query samples are classified via a softmax over distances to class prototypes. Moreover, it shows that using the Euclidean distance function outperforms other methods in this problem.

Another paper helps GPN compute node importance to create class prototypes more efficiently. This paper [29] investigated different methods for estimating node importance in a knowledge graph. The framework consists of score aggregation layers followed by centrality adjustment to score nodes.

For the active learning method, we got the idea from [30] in which a framework is presented to deal with the problem of few-shot learning on graph-structured data and extended for semi-supervised and active learning. The network is trained via both labeled and unlabeled nodes. They considered a fully connected graph in which every edge has a different weight and these weights are given by a learnable similarity kernel. In this graph neural network structure, the trainable adjacency matrix is computed in every layer, and then, a convolution layer is applied. In the active learning experiments, the network can learn to query the label of an unlabeled node which is the most informative one for prediction and can improve the performance of the whole network.

3.2 Proposed model

Our active learning method on graph neural networks for solving a few-shot learning problem is trained episodically according to the meta-learning approach which is defined in [28]. In the training phase, the network is trained on several different tasks. Then, the learned knowledge is generalized to the test phase and classes it has never seen before. The overall architecture of the method is shown in Fig. 1.

Specifically, an N-way, N-Shot learning task is created in each episode:

$$\begin{aligned}
S_t &= \{(\upsilon_1, y_1), (\upsilon_2, y_2), \ldots, (\upsilon_{N\times K}, y_{N\times K})\},\\
Q_t &= \left\{(\upsilon_1^*, y_1^*), \left(\upsilon_2^*, y_2^*\right), \ldots, \left(\upsilon_{N\times M}^*, y_{N\times M}^*\right)\right\},\\
\mathcal{T}_t &= \{S_t, Q_t\}
\end{aligned} \tag{4}$$

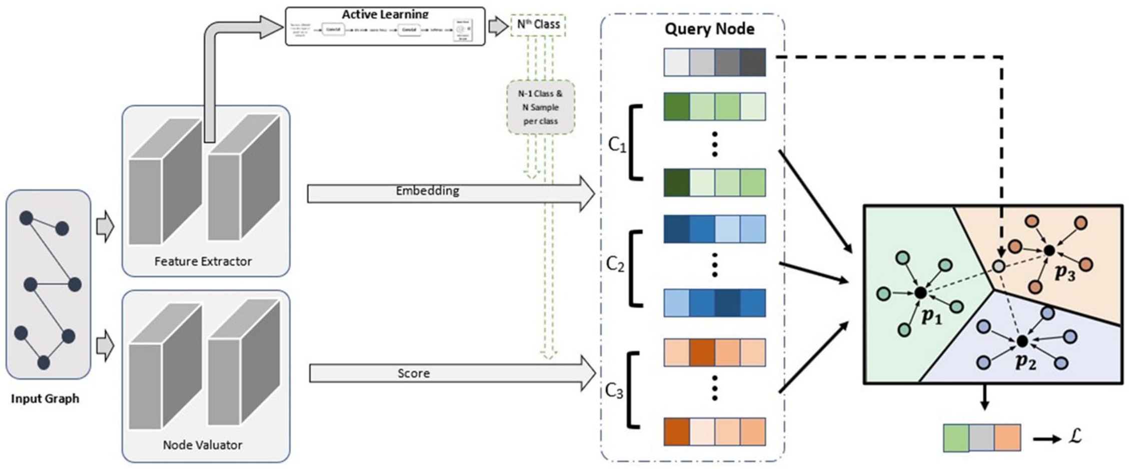

Fig. 1 GPN + AL network architecture

The S_t support set and the Q_t query set in the T_t task are both from training classes. The entire training process is based on a set of meta-learning tasks. The model learns on the training set to minimize the prediction error in the training query set and goes episode by episode to achieve convergence. In this way, the model gradually acquires meta-knowledge so that it can generalize it to the test tasks $T_{\text{test}} = S, Q$ (which are created from new classes in the same way as N-way, N-shot).

3.3 Computing node representation

We have used a two-layer GCN, to map each node into a latent representation with low dimensions. Generally, GCNs follow an idea to gather information from neighbors and calculate the representation of nodes by obtaining and recursively compressing node features from local neighbors. This network is somewhat similar to the GCN described in [4] except for two things: the final layer, in which no softmax function is applied for classification, and its output is node embeddings, the second difference is our active learning network that has been embedded after the first layer. According to [31], if we have the $G = (V, E)$ graph, the GCN input is:

Matrix X with $N \times F$ dimensions, N is the number of nodes, F represents features per node and adjacency matrix A with $N \times N$ dimensions that forms the graph structure. A hidden layer is written in GCN as in formula 4, where $H^0 = X$ and f is the propagation rule.

$$H^i = f\left(H^{i-1}, A\right) \tag{5}$$

As mentioned, the input of a $N \times F$ matrix is from features, where each row corresponds to one of the nodes.

In each layer, the features are collected using the propagation rule to form the features of the next layer (H^i). To obtain the node embeddings, the propagation rule, which acts similarly to the convolution kernel function, is as formula 6:

$$Z = f(x, A) = \left(\hat{A} ReLU\left(\hat{A} X W^{(0)}\right) W^{(1)}\right) \tag{6}$$

where $W^{(0)} \in R^{C \times H}$ is the input weight matrix to the hidden layer with the H attribute and $W^{(1)} \in R^{H \times F}$ is the hidden layer weight matrix to the output layer.

Neural network weights $W^{(0)}$ and $W^{(1)}$ are learned using gradient descent, and in each iteration of the training, the entire dataset is used. Stochasticity in the training process is also done through dropout. In all rules, aggregation occurs at the neighboring nodes, which allows nodes with similar features and labels to communicate with each other and share their features.

3.4 Active learning method

In order to improve the results of classification on few-shot learning tasks, we have used the active learning method described in [30] to select one of the classes and its samples intelligently for the support set. So that in each episode, $N - 1$ classes are selected randomly and the Nth class will be selected using active learning instead of choosing N classes and k samples of each. For this purpose, as illustrated in Fig. 1, after the first layer of the embedding network, the active learning function is called to select the class, which the network is more certain about, from the training classes (excluding the selected classes in that episode), as the most informative class. Then, k samples are selected from the available samples that are most likely to belong to the chosen class. The selected samples' indices will be added to the support set in that episode.

The attention mechanism is used to find the most informative node to obtain its label. The node query operation is performed after the first layer of our embedding network, defined in the previous section, using attention softmax over the graph's unlabeled nodes. To do this, a **g** function is applied that maps each unlabeled node's graph to a scalar value. The g function is implemented by a two-layer neural network (according to formula 7)

$$Attention = Softmax\left(g\left(x^{(1)}_{\{1,\ldots,r\}}\right)\right) \tag{7}$$

It consists of a two-layer, one-dimensional convolutional network that uses the softmax function to obtain the probability that each node belongs to different classes. Then, the sample with the maximum value, about which the network is more certain, is selected as the most valuable node. After that, it obtains the selected sample's label and adds it to the collection. After integrating the new label into the selected node, the information is propagated forward, and this attention part is trained end-to-end along with the rest of the network through the backward propagation of the loss obtained from the neural network's output.

The general structure of the active learning method is shown in Fig. 2.

3.5 Computing prototypes of classes

By learning the node representations from the first network, the next step is to calculate the representation of each class using the labeled nodes in the support set. This section follows the paradigm of prototypical networks [19], in which the nodes of each class cluster form a specific prototype. Class

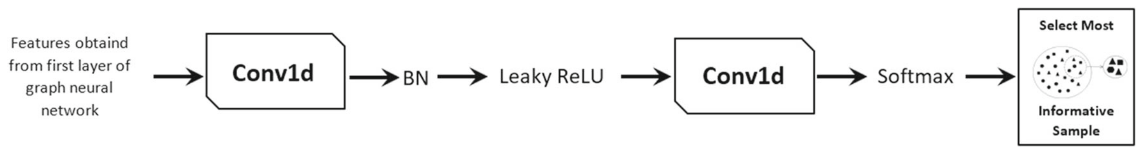

Fig. 2 Proposed structure for active learning using a two-layer neural network

prototypes can be calculated using formula 8:

$$P_c = \frac{1}{|S_c|} \sum_{i \in S_c} Z_i \tag{8}$$

where S_c are the labeled instances of c class and Z_i are the learned features of the nodes from the representation network, and the prototype of each class (P_c) is calculated by averaging the total embeddings of the nodes belonging to that class.

It is worth noting that by using active learning method to select one of the classes, the network learns how to select a much more valuable instance in each iteration so that the class prototypes will be created more accurately. Consequently, node classification will be done more precisely as well as minimizing the amount of network loss and error.

3.6 Determine the importance of each node

Despite the simplicity of this method, which means the direct use of the average embedding vectors of support samples as a prototype, it may not provide promising results for few-shot learning problem [28]. In fact, it ignores that each node has different importance in the network and makes the FSL (Few-shot Learning) model very noise-sensitive since the labeled data are very limited. Therefore, the way class prototypes are created is essential to build a robust and effective FSL model.

To identify the value of each labeled node, it is considered that the node importance is closely related to its neighbors' importance. Accordingly, we have followed the simplified node valuator method (as shown in Fig. 1) to estimate the importance degree of the node through it, which is again calculated using a two-layer GCN considering a fully connected layer at the end that map each node to a scalar value. Therefore, the network output is a score for each node, which is represented by S_i^L.

Since the importance of each node is positively related to its centrality in the graph, and each node's degree is a measure of its centrality and popularity, a node's centrality in the graph is calculated by formula 9:

$$C(i) = \log(\deg(i) + \epsilon) \tag{9}$$

(ϵ is a small constant value) To calculate the final importance of each node, the calculated centrality of formula 11 is applied to the output obtained from the node valuator discussed earlier; then, the sigmoid non-linear function is applied to it according to formula 10.

$$\tilde{S}_i = sigmoid\Big(C(i) \cdot S_i^L\Big) \tag{10}$$

So, the significance of the labeled samples in the support set is adjusted; by doing so, the prototypes of the classes are represented much stronger. The significance of nodes is applied to the support set's embeddings, and then, the prototype of each class is calculated from the set of new embeddings obtained.

3.7 Training

The network that obtains node embeddings, the active learning network, and the node valuator network is trained end to end with the rest of the model. At the end of the training, to classify and calculate the probability that each of the query set samples (V_i) belongs to the desired classes, the squared Euclidean distance of embeddings of the sample set (Z_i) (which is from the resulting of the first network) is obtained from the class prototypes (P_c), and then, the softmax function is applied (formula 11):

$$P\left(c|\upsilon_i^*\right) = \frac{\exp\left(-d\left(Z_i^*, P_c\right)\right)}{\sum_{C'} \exp\left(-d\left(Z_i^*, P_{C'}\right)\right)}, \tag{11}$$

($d(0)$) is the distance function. After calculating the network loss function, the backward propagation and optimization are calculated. In an episodic training context, the goal of each task is to minimize the classification errors between model prediction on the query set and the actual samples label. The classification error is calculated by formula 12:

$$\mathcal{L} = -\frac{1}{N \times M} \sum_{i=1}^{N \times M} log\, P(y_i^* | \upsilon_i^*) \tag{12}$$

After training a significant number of meta-training tasks, its generalized performance in the test phase is measured. In each test episode, the predictor model generated by our model is used to classify each node of the query set into the

Table 1 Statistics of the evaluation datasets

Dataset	Nodes	Edges	Attributes	Labels
Amazon—electronics	42,318	43,556	8669	167
Amazon—clothing	24,919	91,680	9034	77
DBLP	40,672	288,270	7202	137

most probable class (according to formula 13):

$$\hat{y}_i^* = argmax_C P(c|\upsilon_i^*) \tag{13}$$

4 Experimental results

4.1 Datasets

In this section, we evaluate the performance of our proposed method on different datasets.

Amazon-electronics dataset [32] spanning May 1996–July 2014. To use this dataset in our model, each node represents an Amazon product in electronics category and features of nodes are derived from the product description. And a complementary relationship (bought-together) between products is used to create the edges.

From this dataset, we select 90/37/40 classes for training/validation/testing. For the input graph of our model, product descriptions (on which the bag of words model is applied) form the features matrix, the adjacency matrix uses the "also_bought" values between products that were bought together to create links between nodes and the labels of products are given by the low-level categories in the metadata such as digital camera and monopod.

Amazon-Clothing, Shoes and Jewelry dataset [32] ranging from May 1996–July 2014 is also similar to the previous dataset in which each node is a product, its description on Amazon website is used to create features by applying bag of words model and low-level category is related to the class of nodes. But for creating links between nodes, substitution relation ("also_viewed" value), which means those similar products recommended to you to buy instead, is considered. Here, we use 40/17/20 classes for training/validation/testing.

DBLP dataset [33] (version 11): It is a dataset of citation networks in which each paper represents a node and its citation to other papers defines links among them. Applying Bag of word model on papers' abstract creates features of a node and publication venue, e.g., journal or conference is used as the labels of each node (only venues which have lasted at least 20 years). We use 80/27/30 node classes for training/validation/test.

In all datasets, only classes with 100 to 1000 nodes are kept and others are excluded. The generated graph's information from the dataset is summarized in Table 1.

4.2 Setup

The settings of representation and evaluator module are the same as the suggestion of their original paper, a two-layer GCN with 16 and 32 dimensions and ReLU activation function with a learning rate of 0.005, 0.0005 of weight decay, (random) dropout amount of 0.5, and Adam optimizer are used. In our active learning network, we use 2 one-dimensional convolution networks with (1×1) filter size which the features from the first layer of representation network are given as input, and the number of classes in each dataset is the output, followed by batch normalization, Leaky ReLU function and softmax.

A total of 300 episodes are considered in the training process. For the test also, 50 extra-test tasks, similar to what we created in the training task, are randomly selected from the test-related classes.

We evaluate the performance of our active learning GPN model on Amazon electronic and clothing datasets as well as DBLP dataset in four FSL classification tasks of: Since the usual values of the N parameter in related studies vary from 5, 10, 20 and for k parameter 1, 3 and 5 are common. So, we select 5 and 10 for the number of our classes and 3 and 5 for shots to assess our model on few-shot tasks problem. 5way-3shot, 5way-5shot, 10way-3shot, and 10way-5shot (in each task, the size of the query set is equal to the number of shots in support set), and presented the results in Table 2 (Performances of other models are cited from our base paper, GPN [28]).

4.3 Comparison

In order to compare, the two common criteria accuracy (ACC) and Micro-F1 (F1) are used for performance evaluation. We compare our model against related baseline methods:

Deepwalk [34]: uses local information obtained from truncated random walks as input to learn latent representations of nodes in a graph.

Table 2 Comparison of the performance of different algorithms on the Amazon-electronic dataset

Model	5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk	23.5	22.2	26.1	25.7	14.7	12.9	16.0	14.7
Node2vec	25.5	23.7	27.1	24.3	15.1	13.1	17.7	15.5
GCN	53.8	49.8	59.6	55.3	42.3	38.4	47.4	48.3
SGC	54.6	53.4	60.8	59.4	43.2	41.5	50.0	47.6
PN	53.5	55.6	59.7	61.5	39.9	40.0	45.0	44.8
MAML	53.5	52.1	59.0	58.3	37.4	36.1	43.4	41.3
Meta-GNN	63.2	61.5	67.9	66.8	58.2	55.8	60.8	60.1
GPN	64.6	61.8	70.9	70.6	58.6	56	62.4	63.7
GPN + AL	**66.3**	**62**	**73**	**72**	**60.1**	**57.6**	**65.9**	**64.9**

Node2vec [35]: It generalizes Deepwalk by exploring network neighborhoods flexible and controllable by designing biased random walk procedure.

GCN [4]: GCN model uses an efficient layer-wise propagation rule that is based on a first-order approximation of spectral convolutions on graphs to learn representations of nodes.

SGC [14]: SGC simplifies GCN through removing nonlinearities and collapsing the resulting function into a single linear transformation.

PN [19]: It learns a metric space in which classification can be performed by computing distances to prototype representations of each class for few-shot learning.

MAML [16]: represents a model to enable fast learning of new tasks for meta-learning.

Meta-GNN [36]: This model incorporates the meta-learning approach into graph neural networks, providing the capability of well generalizing to new classes that have never been encountered before, with very few samples.

GPN [28]: It proposes a novel paradigm to solve few-shot learning problem by using GCNs to learn class prototypes.

The results of Tables 2, 3 and 4 show that the proposed GPN + AL approach, which its results are bolded, has achieved the best performance in all FSL tasks compared to other models. (The results of our model are the average of all performances after 30 runs.) In general, DeepWalk and node2vec fall behind other methods in FSL Tasks. These Random Walk-based random methods rely on large amounts of labeled data for acceptable performance. Similarly, GNN-based methods cannot achieve competitive results in the mentioned issues. Conventional GNN models have been developed for the semi-supervised node classification and simply suffer from over-fitting with a small number of labeled samples.

MAML and PN also perform poorly in such tasks. The main reason is that these methods cannot maintain the dependency between the nodes to learn their representation.

By integrating the approach of meta-learning into graph neural networks, Meta-GNN has made significant advances over other basic methods in FSL classification in most cases [28]. GPN has surpassed the previous methods by providing a method based on encoder and evaluator networks. Finally, our model by adding active learning method demonstrates significantly higher performance in all cases. The observations show that our proposed method has competitive results than the best model, GPN.

Additionally, we can further observe that by increasing the class size from 5 to 10, the performance of all few-shot studied models decreases. The GPN model performed better than previous ones due to the impact of evaluating node values to learn class prototypes. Furthermore, the effect of active learning addition on the performance of that model is evident owing to the creation of more accurate class prototypes, which have achieved better results than other models. The results in Tables 2, 3 and 4 also illustrate that the performance of all models increases as the number of shots grows from 3 to 5. Also, among prior listed classification methods, the GPN model has provided better performance by increasing the number of shots due to the need for more support set samples to create more accurate prototypes of classes, and our proposed model has also helped this by selecting more useful samples and has significantly improved results. In the next section, Kruskal–Wallis statistical test is carried on ensuring that the proposed model performance has a statistically significant difference.

4.3.1 Statistical test

To demonstrate the contrast in performance of the proposed model, we have computed the Kruskal–Wallis statistical test and Bonferroni post hoc analysis on accuracy and F1 results of models with different N-way K-shot tasks on three datasets.

Table 3 Comparison of the performance of different algorithms on the Amazon-clothing

Model	5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk	36.7	36.3	46.5	46.6	21.3	19.1	35.3	32.9
Node2vec	36.2	35.8	41.9	40.7	17.5	15.1	32.6	30.2
GCN	54.3	51.4	59.3	56.6	41.3	37.5	44.8	40.3
SGC	56.8	55.2	62.2	61.5	43.1	41.6	46.3	44.7
PN	53.7	53.6	63.5	63.7	41.5	41.9	44.8	46.2
MAML	55.2	54.5	66.1	67.8	45.6	43.3	46.8	45.6
Meta-GNN	74.1	73.6	77.3	77.5	61.4	59.7	64.2	62.9
GPN	75.4	74.7	78.6	79.0	65.0	63.2	67.7	68.9
GPN + AL	**80.5**	**79.4**	**83.4**	**82.5**	**67.4**	**65.5**	**72.1**	**71**

Table 4 Comparison of the performance of different algorithms on the DBLP

Model	5-way 3-shot		5-way 5-shot		10-way 3-shot		10-way 5-shot	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk	44.7	43.1	62.4	60.4	33.8	30.8	45.1	43.0
Node2vec	40.7	38.5	58.6	57.2	31.5	27.8	41.2	39.6
GCN	59.6	54.9	68.3	66.0	43.9	39.0	51.2	47.6
SGC	57.3	54.7	65.0	62.1	40.2	36.8	50.3	46.4
PN	37.2	36.7	43.4	44.3	26.2	26.0	32.6	32.8
MAML	39.7	39.7	45.5	43.7	30.8	25.3	34.7	31.2
Meta-GNN	70.9	70.3	78.2	78.2	60.7	60.4	68.1	67.2
GPN	74.5	73.9	80.1	79.8	62.6	62.6	69.0	69.4
GPN + AL	**79**	**76.9**	**81.8**	**81.1**	**66**	**63.0**	**71.2**	**69.8**

This paired comparison test is presented in Table 5. Regarding the p values we can conclude that our proposed method had a significant distinction with Deep Walk, Node2vec, GCN, PN, and MAML in various tests. It can also be obvious that although its comparisons with SGC, Meta-GNN, and GPN were not dramatically different, the proposed model surpassed them as well.

4.4 Loss analysis

Moreover, we also compare our model in terms of loss rate with GPN which has the highest performance among others in similar conditions and settings in 5way-5shot stance (in which best results in all datasets were obtained). The comparison results of the two models are given in Table 6 and Fig. 3.

This table illustrates that our model has lower loss rate in three datasets against GPN model. Therefore, it can be concluded that the more intelligently classes and their samples in the support set are selected, the more accurate the prototypes of embeddings will be, leading to an overall improvement in model's prediction. Consequently, makes it more robust and reliable for solving classification problems with few shots.

4.5 Parameters analysis

To analyze the models' sensitivity to the number of classes (N-way), the size of the support set (K-shot), and the size of the query set, we have performed experiments on our model and GPN, which will be discussed in this section.

4.5.1 Effect of class size (N-way)

First, we analyze the effect of class size on tasks that are controlled by the parameter N. The performance changes in the models in terms of accuracy (ACC) by adjusting the different values of N are presented in Fig. 4.

As can be seen, by increasing the class size, the performance of models decreases as more classes lead to the prediction of more types of nodes, increasing the difficulty of classifying with a few-shot approach.

Table 5 Pairwise comparisons of models

ACC

	5-way 3-shot			5-way 5-shot			10-way 3-shot			10-way 5-shot		
	H statistic	Std. Error	p-value	H statistic	Std. Error	p-value	H statistic	Std. Error	p-value	H statistic	Std. Error	p-value
Node2vec-GPN + AL	−20/667	−3/189	0/001	−20/667	−3/189	0/001	−20/667	−3/189	0/001	−20/833	−3/216	0/001
Deep Walk-GPN + AL	−20/333	−3/138	0/002	−17/667	−2/726	0/006	−20/333	−3/138	0/002	−18/667	−2/881	0/004
PN-GPN + AL	−16/167	−2/495	0/013	−14/667	−2/263	0/024	−15/333	−2/366	0/018	−17	−2/624	0/009
MAML-GPN + AL	−14/5	−2/238	0/025	−14/667	−2/263	0/024	−13/667	−2/109	0/035	−16	−2/47	0/014
GCN-GPN + AL	−10/333	−1/595	0/111	−12	−1/852	0/064	−10/667	−1/646	0/1	−10/833	−1/672	0/094
SGC-GPN + AL	−9	−1/389	0/165	−11	−1/697	0/09	−10/333	−1/594	0/111	−9/667	−1/492	0/136
Meta-GNN-GPN + AL	−3/333	−0/514	0/607	−3/667	−0/566	0/572	−3/333	−0/514	0/607	−3/667	−0/566	0/571
GPN-GPN + AL	−1/667	−0/257	0/797	−1/667	−0/257	0/797	−1/667	−0/257	0/797	−2/333	−0/36	0/719

F1

	5-way 3-shot			5-way 5-shot			10-way 3-shot			10-way 5-shot		
	H statistic	Std. Error	p-value	H statistic	Std. Error	p-value	H statistic	Std. Error	p-value	H statistic	Std. Error	p-value
Node2vec-GPN + AL	−21	−3/24	0/001	−20/333	−3/138	0/002	−20/333	−3/138	0/002	−21	−3/241	0/001
Deep Walk-GPN + AL	−20/333	−3/138	0/002	−18	−2/778	0/005	−20	−3/086	0/002	−19/333	−2/984	0/003
PN-GPN + AL	−12/667	−1/955	0/051	−13/167	−2/032	0/042	−12	−1/852	0/064	−14/667	−2/263	0/024
MAML-GPN + AL	−14	−2/16	0/031	−13/667	−2/109	0/035	−13/667	−2/109	0/035	−16/333	−2/521	0/012
GCN-GPN + AL	−13	−2/006	0/045	−14	−2/161	0/031	−12/333	−1/903	0/057	−10/833	−1/672	0/095
SGC-GPN + AL	−6/667	−1/029	0/304	−11/5	−1/775	0/076	−10/667	−1/646	0/1	−10/833	−1/672	0/095
Meta-GNN-GPN + AL	−3/333	−0/514	0/607	−3/667	−0/566	0/571	−3	−0/463	0/643	−4/333	−0/669	0/504
GPN-GPN + AL	−1/667	−0/257	0/797	−1/667	−0/257	0/797	−1	−0/154	0/877	−1/667	−0/257	0/797

Table 6 Comparison of the loss of two algorithms on the different datasets

Model/dataset	Electronics	Clothing	DBLP
GPN	1.0237124	0.62608	0.6786
GPN + AL	0.7767061	0.53433	0.5269

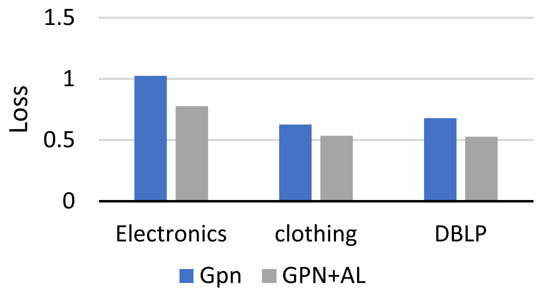

Fig. 3 Loss comparisons of GPN and GPN + AL on different datasets. (5-way 5-shot)

4.5.2 Effect of support set size (k-shot)

Next, we examine the effect of the sample size of the support set, which is represented by the K parameter. We have reported the results in terms of accuracy (ACC) in Fig. 5. According to the diagram, it can be clearly seen that the performance of models increases as the value of k grows, indicating that a larger support set can produce better prototypes.

4.5.3 Effect of query set size (M)

In this section, we have used 5way-5shot tasks. Then, we have changed the number of query set samples from each class and reported the relevant results in Fig. 6. From the reported

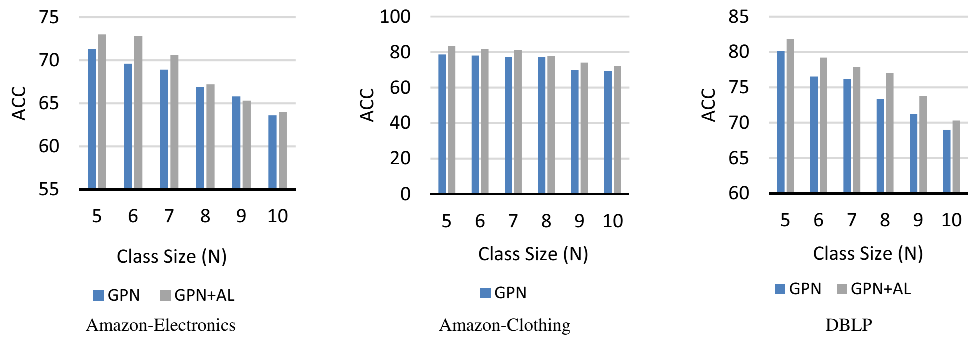

Fig. 4 The diagram of the class size effect in Nway-5shot task

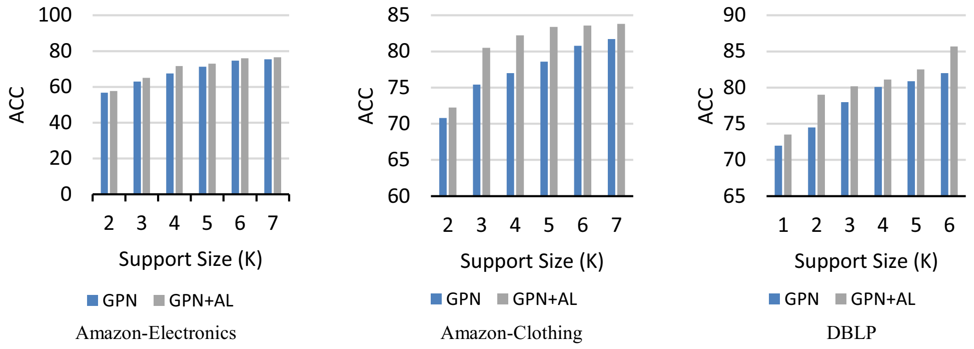

Fig. 5 The diagram of the shot size effect in 5way-Kshot task

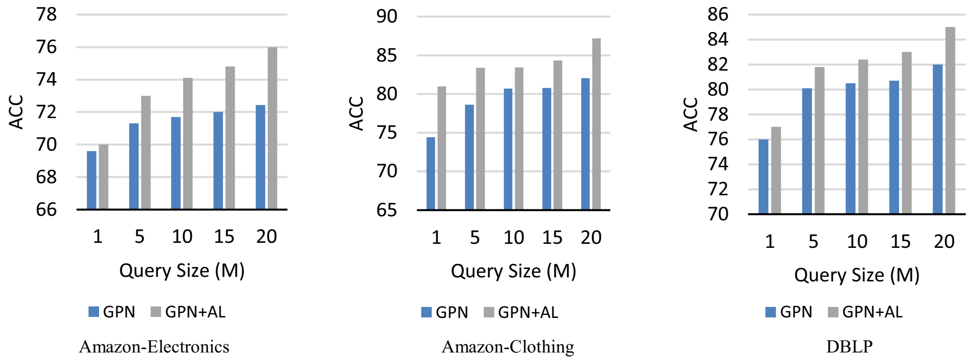

Fig. 6 The diagram of the query size effect in 5way-5shot, M query task

results, we can see that increasing the size of the query set increases the models' accuracy and when M = 20, the highest performance of the models is obtained. For example, our proposed model reached an accuracy of 87.2% in this situation on Amazon-Clothing dataset because an episodic learning model could match the knowledge gained from meta-learning tasks with larger query sets and gain a better generalization ability to the desired tasks [28].

The observations show that the proposed method has a better performance than the GPN model and it is more robust and reliable for solving classification problems with few shots. Therefore, it can be concluded that the more intelligently classes and their samples in the support set are selected, the more accurate the prototypes of embeddings will be, leading to an overall improvement in model prediction.

5 Conclusion

In this paper, we presented an efficient way to classify data with a small number of samples. According to what is described in the third section, this method includes two powerful networks of convolutional graphs, one of which is to obtain node embeddings and the other is to determine the importance of nodes according to their centrality and connection with neighboring nodes. Also, another convolutional network is built into the first network to intelligently select samples. So, it chooses the most valuable sample, about which the network is more certain, from unlabeled data, to minimize network loss. The score obtained from the second network affects the output of the first network to create a class prototype, and in the final step, according to the distance between the embedding of the sample we want to classify and the class prototype, we predict the sample label. Then, we generalize the knowledge obtained from these steps to the test phase for classifying new classes. Experimental results show that the proposed method of using active learning has made remarkable progress compared to the basic method and has led to increased accuracy and reduced network loss. In future studies, other few-shot learning methods, discussed in the second section, on convolutional graph networks can be used, and samples can be selected intelligently through active learning or making changes in the calculating method of the classes' prototypes and evaluator network, which has a significant impact on the results and the examination of whether they will perform better than the method discussed in this paper or not. The performance of the proposed method can also be tested on a larger dataset.

References

1. Haider, S.: *Few Shot Learning for Text*. M.S thesis. Faculty of Media, Bauhaus-Universität, Weimar, Germany (2020)
2. Estay, B. [Online]. https://www.bigcommerce.com/blog/online-shopping-statistics/#ecommerce-is-growing-every-day. Accessed 1 Sept 2021
3. [Online]. https://catsy.com/blog/product-categorization/. Accessed 1 Sept 2021
4. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). arXiv:1609.02907
5. Wang, Z., Gao, H., Ji, S.: Large-scale learnable graph convolutional networks. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1416–1424. ACM (2018)
6. Ying, Z., Leskovec, J., Hamilton, W.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, December 2017 (NIPS'17), pp. 1025–1035 (2017)
7. Ma, T., Xiao, C., Chen, J.: Fastgcn: fast learning with graph convolutional networks via importance sampling. In: *Proceedings of the 7th International Conference on Learning Representations* (2018)
8. Liu, X., Si, S., Li, Y., Bengio, S., Hsieh, C.J., Chiang, W.L.: Clustergcn: an efficient algorithm for training deep and large graph convolutional networks (2019). arXiv:1905.07953
9. Kapoor, A., Perozzi, B., Lee, J., Abu-El-Haija, S.: NGCN: multi-scale graph convolution for semi-supervise node classification (2018). arXiv:1802.08888
10. Zhuang, C., Ma, Q.: Dual graph convolutional networks for graph-based semi-supervised classification. In: *Proceedings of the 2018 World Wide Web Conference*, pp. 499–508 (2018)
11. Zhu, J., Song, L., Chen, J.: Stochastic training of graph convolutional networks with variance reduction. In: *International Conference on Machine Learning*, pp. 941–949 (2018)
12. Han, Z., Wu, X.M., Li, Q.: Deeper insights into graph convolutional networks for semi-supervised learning. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (2018)
13. Verma, S., Zhang, Z.L.: Stability and generalization of graph convolutional neural networks (2019). arXiv:1905.01004
14. Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K., Wu, F.: Simplifying graph convolutional networks. In: *International Conference on Machine Learning*, pp. 6861–6871 (2019)
15. Han, Y.: *A New Method to Solve Same-Different Problems with Few-Shot Learning*. M.S thesis, Western University, Ontario, Canada (2019)
16. Abbeel, P., Levine, S., Finn, C.: Model-agnostic meta-learning for fast adaptation of deep networks. In: *Proceedings of the International Conference on Machine Learning* (2017)
17. Ravi, S., Larochelle, H.: Optimization as a model for few-shot learning. In: *ICLR* (2017)
18. Blundell, C., Lillicrap, T., Wierstra, D., Vinyals, O.: Matching networks for one shot learning. In: *Advances in Neural Information Processing Systems*, pp. 3630–3638 (2016)
19. Swersky, K., Zemel, R., Snell, J.: Prototypical networks for few-shot learning (2017). arXiv:1703.05175
20. Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M., Sung, F.: Learning to compare: relation network for few-shot learning (2017). arXiv:1711.06025
21. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: *ICML Deep Learning Workshop*, vol. 2 (2015)
22. Sörsäter, M.: *Active Learning for Road Segmentation using Convolutional Neural Networks. M.S thesis*. Department of Electrical Engineering, Linköping University, Linköping, Sweden (2018)
23. Angluin, D.: Queries and concept learning. Mach. Learn. **2**(4), 319–342 (1988)
24. Cohn, D., Ladner, R., Atlas, L.: Training connectionist networks with queries and selective sampling. In: *Advances in Neural Information Processing Systems*, pp. 566–573 (1990)
25. Atlas, L., Ladner, R., Cohn, D.: Improving generalization with active learning. Mach. Learn. **15**(2), 201–221 (1994)
26. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: *SIGIR'94*, pp. 3–12. Springer (1994)
27. Settles, B.: *Active Learning Literature Survey*. Technical report, University of Wisconsin-Madison Department of Computer Sciences (2009)
28. Wang, J., Li, J., Shu, K., Liu, C., Liu, H., Ding, K.: Graph prototypical networks for few-shot learning on attributed networks. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020)
29. Kan, A., Dong, X.L., Zhao, T., Faloutsos, C., Park, N.: Estimating node importance in knowledge graphs using graph neural networks. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019)
30. Garcia, V., Bruna, J.: Few-shot learning with graph neural networks. In *Proceedings of the International Conference on Learning Representations* (2018)
31. Kipf, T. [Online]. http://tkipf.github.io/graph-convolutional-networks. Accessed 23 July 2021
32. Pandey, R., Leskovec, J., McAuley, J.: Inferring networks of substitutable and complementary products. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015)

33. Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z., Tang, J.: Arnetminer: extraction and mining of academic social networks. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2008)
34. Al-Rfou, R., Skiena, S., Perozzi, B.: Deepwalk: online learning of social representations. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2014)
35. Leskovec, J., Grover, A.: node2vec: Scalable feature learning for networks. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016)
36. Cao, C., Zhang, K., Trajcevski, G., Zhong, T., Geng, J., Zhou, F.: Meta-gnn: on few-shot node classification in graph meta-learning. In: *Proceedings of the ACM International Conference on Information and Knowledge Management* (2019)