Feature-Attention Graph Convolutional Networks for Noise Resilient Learning

Min Shi, *Student Member, IEEE*, Yufei Tang, *Member, IEEE*, Xingquan Zhu, *Senior Member, IEEE*, Yuan Zhuang, *Member, IEEE*, Maohua Lin, and Jianxun Liu

***Abstract*—Noise and inconsistency commonly exist in real-world information networks, due to the inherent error-prone nature of human or user privacy concerns. To date, tremendous efforts have been made to advance feature learning from networks, including the most recent graph convolutional networks (GCNs) or attention GCN, by integrating node content and topology structures. However, all existing methods consider networks as error-free sources and treat feature content in each node as independent and equally important to model node relations. Noisy node content, combined with sparse features, provides essential challenges for existing methods to be used in real-world noisy networks. In this article, we propose feature-based attention GCN (FA-GCN), a feature-attention graph convolution learning framework, to handle networks with noisy and sparse node content. To tackle noise and sparse content in each node, FA-GCN first employs a long short-term memory (LSTM) network to learn dense representation for each node feature. To model interactions between neighboring nodes, a feature-attention mechanism is introduced to allow neighboring nodes to learn and vary feature importance, with respect to their connections. By using a spectral-based graph convolution aggregation process, each node is allowed to concentrate more on the most determining neighborhood features aligned with the corresponding learning task. Experiments and validations, w.r.t. different noise levels, demonstrate that FA-GCN achieves better performance than the state-of-the-art methods in both noise-free and noisy network environments.**

***Index Terms*—Feature attention, graph neural networks (GNNs), graphs, noise, representation learning.**

I. Introduction

MANY real-world applications involve knowledge mining and analysis of networks or graph-based data, such as citation networks, social networks, telecommunication networks, biological networks, etc., which are often collected from noisy channels with erroneous/inconsistent labels or features [1]. In order to carry out pattern mining from networks, such as community detection [2], node classification [3], link prediction [4], etc., network representation learning (or embedding learning) [5] is commonly used to construct features to represent nodes for learning.

To capture node relations, early network embedding learning mainly focuses on topology features [2], [6], where nodes sharing similar topology structures are enforced to have close feature representation. For example, two scholarly publications citing similar literature in a citation network would be represented by similar feature vectors [7]. Likewise, two users interacting with many common friends in a social network would share similar features in the learned representation space [8]. However, structure-based methods can only model explicit node relations reflected by the network edges, which is inadequate to capture implicit relationships between nodes because of the sparse graph connections. For example, two users in a social network do not have an immediate link not because they are not friends in reality but they might be unaware of each other's existence online. To mitigate this problem, studies propose to embed content information associated with a network to enhance node structures modeling [9], [10]. Indeed, networks with rich textual contents are ubiquitous in the real world, such as the citation network and Wikipedia network where nodes are usually described by substantial texts. In general, content features are able to reveal relationships between nodes aligned with the network structures (e.g., two nodes with many shared content features are highly likely to form a neighborhood) [11], [12], but in a more fine-grained and interpretable fashion, for example, the affinity between two linked nodes can be measured by overlapping content features between each other, other than just a single edge connecting the nodes.

In addition to the above adjacency matrix or random walk-based network embedding learning, recently, the spectral-based graph convolutional networks (GCNs) [13], [14] have shown impressive performance to directly embed graphs with rich content features by a semisupervised node classification training. GCN relies on the assumption that a node tends to

Manuscript received January 2, 2020; revised April 18, 2021 and September 27, 2021; accepted January 7, 2022. This work was supported in part by the National Science Foundation under Grant IIS-1763452, Grant CNS-1828181, and Grant OAC-2017597. This article was recommended by Associate Editor H. Yin. *(Corresponding author: Yufei Tang.)*

Min Shi, Yufei Tang, and Xingquan Zhu are with the Department of Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: mshi2018@fau.edu; tangy@fau.edu; xzhu3@fau.edu).

Yuan Zhuang is with the State Key Laboratory of Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430072, China (e-mail: yuan.zhuang@whu.edu.cn).

Maohua Lin is with the Department of Ocean and Mechanical Engineering, Florida Atlantic University, Boca Raton, FL 33431 USA (e-mail: mlin2014@fau.edu).

Jianxun Liu is with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, Hunan, China (e-mail: ljx529@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3143798.

Digital Object Identifier 10.1109/TCYB.2022.3143798

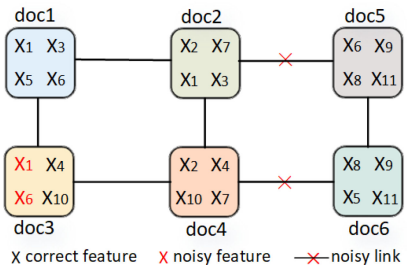


Fig. 1. Example of a document network with noisy content and noisy structure. In the example: 1) noisy content means erroneous feature values in some nodes (e.g., doc_3) and 2) noisy structure means erroneous links between some pairs of nodes (e.g., $<\text{doc}_2, \text{doc}_5>$ have erroneous links since they do not share any common features). Both cases can be understood that node features fail to characterize true structural relationships between nodes.

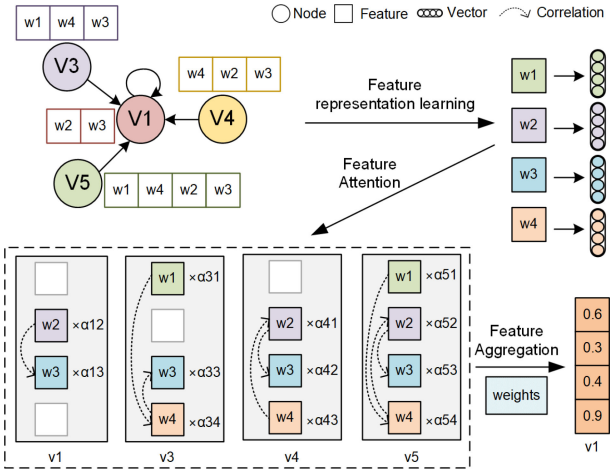


Fig. 2. Illustrative example of the proposed approach: Feature representation is used to explore feature correlation and learn a dense vector for each node feature. Feature attention is used to differentiate feature interaction between each node and its neighboring node features, allowing better feature aggregation for noise resilient embedding learning.

have the same label with its neighbors that is guaranteed by the aggregated features from all neighborhood nodes, where different features are typically treated as independent (e.g., isolated) and equally important (e.g., noise free). However, such a learning mechanism may be challenged by the following two realities. First, in graphs (e.g., citation networks and Wikipedia networks) where node contents are feature sequences, different features are usually not isolated but correlate with each other to reveal a complete semantic context [15], that is, the meaning of a sentence is usually demonstrated not by every single word but by the context of all words having dependencies or correlations with each other. Second, while nodes sharing connections are assumed to have dependencies in their content features, not all features function equally to trigger the interactions between nodes. For instance, although a research publication may have rich text information, such as title and abstract, many words are actually irrelevant (e.g., noises or errors), which are not indicative of its citation relationships with others.

Indeed, existing methods have made significant progress for network embedding learning, but they all consider networks as error-free sources and treat different features in each node content as isolated and equally important while modeling node relations. It is worth noting that a noisy network can be originated from noisy node content and/or noisy graph structure as the example demonstrates in Fig. 1. Both the two types of noise can be roughly understood that node content cannot well explain the graph structure, and vice versa [16]. For instance, in Fig. 1, doc_1 and doc_3 have a link because they share common features X_1 and X_6. But because X_1 and X_6 are noisy features in doc_3, the link between doc_1 and doc_3 is actually a noisy link. In this article, we are interested to address the impact of noisy node content to achieve improved graph embedding learning. The ignorance of the impact of erroneous content, combined with sparse node features, provides essential challenges for existing methods to handle real-world noisy networks. In fact, a recent work has empirically validated various embedding approaches on sparse and noisy knowledge graphs, and concluded that "*embeddings are sensitive to sparse and unreliable data*" [17].

In summary, existing methods are sensitive and ineffective to noise and sparse content mainly because of the following challenges.

1) *Noise Propagation:* When noise is imposed to the node (e.g., incorrect words), it will force existing methods, such as GCN or attention networks, to learn deteriorated weight values, corresponding to noisy features. Such noise propagation directly deteriorates network embedding performance as we will show in Section V.
2) *Feature Dependency and Interaction:* While existing methods have taken node content into consideration, they treat all features independently and equally for embedding learning. In reality, features are dependent and have different interactions with respect to neighboring nodes, and thereby should be correlated and differentiated for learning each node's representation.
3) *Sparsity and Dimensionality:* Most networks are high dimension with sparse node content (e.g., each node only has about 1% features, compared to the entire feature space). Noise impact, in a high dimensionality and sparse content setting, is a more profound challenge because the underlying models are highly vulnerable to errors.

To address aforementioned problems, we propose a novel feature-based attention GCN (FA-GCN) model to perform noise resilient learning for networks with sparse and noisy node content. Fig. 2 shows an illustrative example of our proposed approach. First, we represent each content feature as a dense semantic vector, with feature correlation/dependency being well preserved based on a bidirectional long short-term memory (LSTM) network. In other words, the representation learning for each content feature is dependent on the semantic representations of other features of the node content. Since such feature dependencies can be globally learned and shared across all sparse node contents, we expect to learn more accurate feature representations, compared with existing methods that typically model the sparse features for each network node independently and locally. Learning dependencies between highly correlated features would also help mitigate the influence of noisy features for some individual nodes, because

random noisy features are often not correlated with correct features. Second, to minimize the impact of noisy content features, we introduce an attention layer over the LSTM network to determine the importance of various neighborhood features for modeling pairwise node interactions, aligned with the final node classification task. For example, when modeling the interaction between doc_3 and doc_4 in Fig. 1, the feature-level attention will emphasize more on common features (e.g., X_4 and X_{10}) than other irrelevant features such as the noisy features X_1 and X_6 in doc_3. Since noisy features in a node will receive reduced attentions to minimize its impact, they are less likely to propagate over the entire network disastrously, thereby resulting in a noise-resilient graph embedding learning.

It is worth noting that our work is different from the graph attention networks (GATs) [18], wherein features are modeled as independent and the attention is calculated at the node level. In comparison, we argue that features in a node content could interact with each other to reveal richer and more accurate node semantics, that is, the same word feature may have different meanings under different contexts, and different word features may indicate the same meaning within a similar context. Meanwhile, the node-level attention in GAT assumes that all features in the node contents are contributing equally to edge connections, whereas our feature-level attention enables differentiation of relevant features triggering node interactions in a network. Specifically, our main contributions are as follows.

1) We proposed to model node relations at feature level, where each node interacts with different neighbors and their interactions are attributed to the most influential node features.
2) We proposed to model feature correlations for enhanced node representation learning and classification, where networks contain rich node contents or features, such as word sequences.
3) We proposed a noise-resilient learning framework for networks with sparse and noisy text features. It models feature correlations by a bidirectional LSTM network and meanwhile, conducts differentiable neighborhood features aggregation by a higher attention layer over the LSTM network.

The remainder of this article is organized as follows. Section II discusses related work, followed by problem definition and preliminaries related to the LSTM network and spectral-based GCNs in Section III. The proposed algorithm is described in Section IV, and experiments and comparisons are reported in Section V. Finally, Section VI concludes this article.

II. Related Work

A. Graph Representation Learning

Graph representation learning [5], [19] aims to represent each node of a target network as a low-dimensional vector, such that various downstream analytic tasks can be benefited. Early work in the area mainly focuses on shallow neural models to preserve only the node structures [5]. To capture high-order neighborhood relationships between nodes, DeepWalk [8] performs a random walk process over the entire graph to generate a collection of fixed-length node sequences similar to the natural language sentences. It then explores a widely used neural model Skip-Gram [20] to learn node representations from these node sequences. However, Node2vec [21] demonstrates that DeepWalk has not fully preserved the connectivity patterns between nodes and thus, proposes to combine the breadth-first sampling and depth-first sampling in the random walk process, where the community properties between nodes can be well preserved. LINE [22] is proposed for large-scale network representation learning by preserving the first- and second-order node relations, where the first order is determined by the immediate links and the second-order relation between two nodes is created by their shared neighbors. However, in addition to the complex graph structures that have encoded node relations, graphs are usually associated with rich content information, such as attributes and texts, which also revealed the affinities between nodes [7], [23]. For examples, the relational topic model (RTM) [7] is utilized to model both the documents and link relationships, which assumes that documents with links also have similar topic distributions and semantic representations. TADW [9] leverages the rich texts to enhance the structure-based representation learning based on an equivalent matrix factorization method as the DeepWalk. TriDNR [10] can integrate the node structure, content, and labels in a unified framework, which enforces the node representations to be learned from simultaneously the network structure and text content under the shared model parameters.

B. Graph Neural Networks

Because shallow models have limitations in learning complex relational patterns between nodes [19], there is an increasing number of efforts to explore graph neural networks (GNNs), which take a graph as input and learn node representations by a supervised training process [24]. Recently, inspired by the huge success of convolutional neural networks on grid-like data such as images, a lot of tentative works emerged that seek to adopt a similar convolutional feature extraction process directly on the arbitrarily structured data such as graphs [25], [26]. To date, GCNs [13] have appeared to achieve the state-of-the-art performance in many graph related analytic tasks, which can naturally learn node representations from graph structures and contents. For example, Schlichtkrull *et al.* [14] proposed the relational GCN and applied them to two standard knowledge base completion tasks of link prediction and entity classification. Yao *et al.* [3] proposed the text GCN for text classification, where the text graph is built based on the word co-occurrence and document word relations. Yan *et al.* [27] proposed the spatial–temporal GCN for skeleton-based action recognition. It is formulated on top of a sequence of skeleton graphs, where each node corresponds to a joint of the human body and edges represent the connectivity between joints. In general, while a node could connect with many others in a graph, different neighbors may have different contributions when generating the representation of this node. The GATs [18] were proposed to solve this problem by using a self-attention

strategy to assign large weights on important neighbors for feature aggregations. However, since nodes that interact with each other are usually resulted by fine-grained features [23], node-level attention maybe insufficient to characterize node relations. In comparison, in this article, we focus on the feature-based attention mechanism, which is more flexible and interpretable to model node relations since node interactions can be triggered by varying important features. In addition, we argue that modeling feature correlation is critical for the text described graphs, which is generally ignored by most existing GNN-based models.

C. Noise Resilient Graph Learning

Robust learning with noisy data is a classical yet challenging problem in the machine learning [28], [29] and related fields [30], [31]. Data noise is defined as anything (e.g., feature/input noise and label noise) that obscures the relationship between the features of an instance and classes [32], resulting in suboptimal models and prediction results. Recently, in the graph domain, some efforts have been applied to promote noise-resilient embedding learning [33] and graph application tasks [34]. Qiu *et al.* [33] focused on addressing the noisy or missing links in the graphs and proposed to recover them from node structure similarities, that is, two nodes have some common neighborhoods that tend to have a link. However, the proposed method is under an unsupervised framework, which cannot be generalized to the more popular supervised learning and GNN architectures. Similar research along this direction is to consider adversarial attacks that help to guide designing robust graph learning mechanisms to tolerate the deliberate node feature or structure attacks [35], [36]. Zügner *et al.* [37] proposed to apply unnoticeable perturbations to poison either target graph structure or node features, which can dramatically worsen classification results of the target nodes. Zhu *et al.* [36] proposed a robust GCN model to mitigate the adversarial attacks. Their strategy is to adopt Gaussian distributions as hidden representations of n nodes, which can automatically absorb the effects of adversarial changes in the variances of Gaussian distributions. However, these works mostly target at designing sophisticated attacking mechanisms that cause existing graph embedding algorithms ineffective, or seek to design resilient embedding techniques to combat adversarial attacks applied on a single node or a handful of nodes of interest. Although they are robust to specific attacks, they have many constrains and are not ready to address random and large-scare noises for achieving noise-resilient graph learning.

In comparison, we provide a more general noisy-resilient learning framework that allows for large-scale random noisy features in the node contents. We extend existing GNN models and aim to achieve improved graph embedding results through modeling node feature correlations and feature attentions in the interactions between nodes.

III. Problem Definition and Preliminaries

A. Problem Definition and Motivation

A network is represented as $G = (V, E, \mathbb{C})$, where $V = \{v_i\}_{i=1,\ldots,n}$ is a set of unique nodes and $E = \{e_{i,j}\}_{i,j=1,\ldots,n;\ i\neq j}$ is a set of edges. Let A denote the $n \times n$ adjacency matrix representation of edges with $A_{i,j} = 1$ if $e_{i,j} \in E$ and $A_{i,j} = 0$ if $e_{i,j} \notin E$. Let D and I be the $n \times n$ degree matrix and identity matrix, respectively, where D_{ii} is calculated by $D_{ii} = \sum_j A_{ij}$. For each node v_i, we use cnt_i to denote its content, which is a sequence of word features represented by $\text{cnt}_i = \{w_j\}_{j=1,\ldots,|\text{cnt}_i|}$. For all nodes in G, their contents form the content corpus $\mathbb{C} = \{\text{cnt}_i\}_{i=1,\ldots,n}$. We use $|\mathbb{C}|$ to denote vocabulary (or the number of unique words) in the content corpus. It is common that each node has very sparse content, so $|\text{cnt}_i| \ll |\mathbb{C}|$.

As discussed previously, a noisy network may be caused by either noisy node content or noisy node structures. Both cases can be catastrophically challenging the node relation modeling and network embedding learning. In this article, we focus on network embedding learning with noisy node content. We refer to noise as erroneous node content, where the content of each node cnt_i contains some errors (e.g., erroneous feature values or words). In the sparse and noisy node content setting, our *goal* is to learn good feature representation for each node in the network for classification.

In order to tackle sparse and noisy node content, our motivation, as shown in Fig. 2, is to employ an optimization approach to address the sparsity and errors: 1) learning a dense vector to represent each content word (w_j) and 2) using feature attention to learn weight values for each word, based on the node–node interaction, and then use feature attention to aggregate neighboring nodes for noise resilient representation learning for each node.

As shown in Fig. 2, node v_1 has three neighbors (v_3, v_4, and v_5), each containing some content words. Our first learning objective is to learn a dense feature vector for each word (w_j). Then, feature attention is used to learn weight values α_{ij}, which quantify node v_1's feature-level interactions with respect to neighbor v_i on word w_j. After that, feature aggregation is used to aggregate all v_1's neighbors to learn a good feature representation for v_1. It is worth noting that the learning of feature/word representation and node representation is carried out simultaneously under a unified optimization goal, as we will detail in (28).

B. Long Short-Term Memory

To learn vector representation for words, LSTM networks [38] have achieved significant success [39], [40], thanks to its recurrent learning capacity on sequential data like text. In LSTM, features are not independently modeled but can interact with each other through the memory and state transmission mechanisms. When two reverse-order LSTM networks are combined, each feature is allowed to semantically correlate with any other feature within the same sequence. Fig. 3 shows the interior structure of an LSTM unit/cell [38], where f_t, i_t, and o_t are the forget, input, and output gates, respectively. h_t denotes the cell output at time t, and c_t is the global cell state that enables the sharing of different cell outputs throughout the LSTM network. Features are usually sequentially fed into the LSTM network, where

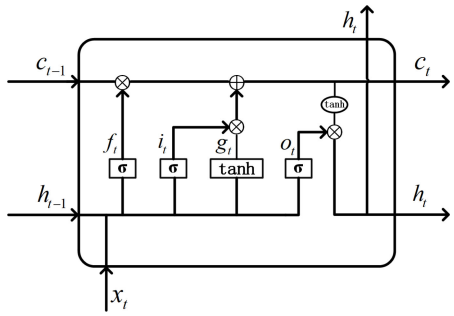


Fig. 3. Structure of an LSTM unit/cell.

the parameters for a feature at time t are updated by

$$f_t = \sigma\left(w_{xf}x_t + w_{hf}h_{t-1} + b_f\right) \tag{1}$$
$$i_t = \sigma\left(w_{xi}x_t + w_{hi}h_{t-1} + b_i\right) \tag{2}$$
$$g_t = \tanh\left(w_{xg}x_t + w_{hg}h_{t-1} + b_g\right) \tag{3}$$
$$f_t = f_t c_{t-1} + i_t g_t \tag{4}$$
$$o_t = \sigma\left(w_{xo}x_t + w_{ho}h_{t-1} + b_o\right) \tag{5}$$
$$h_t = o_t \tanh(c_t) \tag{6}$$

where $w_{x*} = \{w_{xf}, w_{xi}, x_{xg}, w_{xo}\}$ and $w_{h*} = \{w_{hf}, w_{hi}, x_{hg}, w_{ho}\}$ are weight parameters for the corresponding gates, and b_f, b_i, b_g, and b_o are their biases, respectively.

C. Graph Convolutional Networks

GCN is an efficient variant of the convolutional neural networks operating directly on graphs [13] by encoding both graph structures and node features. Given a network $G = (V, E, X)$, which has n vertices and each node has d_o dimension feature values ($X \in \mathbb{R}^{n\times d_o}$ denotes the feature value matrix), GCN intends to learn low-dimensional node representations though a convolutional learning process. In general, with one convolutional layer, GCN is able to preserve the first-order neighborhood relations between nodes, where each node is represented as an l-dimension vector, and node feature matrix $X^{(1)} \in \mathbb{R}^{n\times l}$ is computed by

$$X^{(1)} = \rho\left(\tilde{A}X^{(0)}W_0\right) \tag{7}$$

where $\tilde{A} = D^{-(1/2)}(I+A)D^{-(1/2)}$ is the normalized symmetric adjacency matrix, $X^{(0)} \in \mathbb{R}^{n\times d_o}$ is the initial input feature matrix of X, and $W_0 \in \mathbb{R}^{d_o\times l}$ is a weight matrix for the first convolutional layer. ρ is an activation function such as the *ReLU* represented by $\rho(x) = \max(0, x)$. If it is necessary to encode higher order (e.g., k-hop) neighborhood relationships, one can easily stack multiple GCN layers, where the output node features of the jth ($0 \le j \le k$) layer are computed by

$$X^{(j+1)} = \rho\left(\tilde{A}X^{(j)}W_j\right) \tag{8}$$

where $W_j \in \mathbb{R}^{d_o\times d_o}$ (or $W_j \in \mathbb{R}^{d_o\times l}$ if it is the last layer) is the weight matrix for the jth layer.

IV. Proposed Approach

For existing GCN-based methods, they model node features as independent [e.g., using one-hot representation of content features where each element in the feature matrix X indicates whether a corresponding feature (e.g., word) appears or not], making these methods sensitive to noise and sparse input.

In order to tackle high dimensional, sparse, noisy node content, we propose to represent each single node feature as a dense semantic vector and various features can influence each other by interactions of their semantic vectors during learning. In addition, each feature will be assigned with an importance weight, which allows each node to aggregate important neighborhood features for representation learning.

The proposed FA-GCN model for the above two purposes is shown in Fig. 4. First, the feature representations are learned by a bidirectional LSTM network, where feature correlations can be preserved. Then, with an attention layer on the top, each feature is weighted based on its importance to the final node classification task. In this article, we investigate the effectiveness of two types of feature attention mechanisms that either consider a self-transformation process or introduce a context-aware bilinear term. Finally, each node dynamically aggregates weighted sum of neighborhood features to form its representation. The feature representations and node representations are integrally learned and could enhance each other to optimize a collective classification loss at the end of this framework.

A. Feature Representation Learning

Network node features contain rich semantics and they frequently correlate with each other to trigger complex node connections in a graph. For example, a word feature may have different semantics when correlating with different words under different sentence contexts and these fine-grained semantics could help differentiate the links of the corresponding node with its different neighbors.

As shown in Fig. 4, we use a bidirectional LSTM network to learn representation of each feature from the content corpus $\mathbb{C}$. Let the content features of node v_m be represented by $\text{cnt}_m = \{w_j\}_{j=1,\ldots,|\text{cnt}_m|}$, and we initialize the representations (e.g., with dimension d_i) of these features in the input layer with following a uniform range distribution. Assume the input semantic vector for feature w_j at time t is represented by $\text{vec}_{w_j} \in \mathbb{R}^{d_i}$, and it then undergoes a series of nonlinear transformations in temporal order formulated by

$$\overrightarrow{f}_{t,w_j} = \sigma\left(\mathbf{W}_f\mathbf{X}_t + b_f\right) \tag{9}$$
$$\overrightarrow{i}_{t,w_j} = \sigma\left(\mathbf{W}_i\mathbf{X}_t + b_i\right) \tag{10}$$
$$\overrightarrow{g}_{t,w_j} = \tanh\left(\mathbf{W}_g\mathbf{X}_t + b_g\right) \tag{11}$$
$$\overrightarrow{c}_{t,w_j} = \overrightarrow{f}_{t,w_j}\overrightarrow{c}_{t-1,w_{j-1}} + \overrightarrow{i}_{t,w_j}\overrightarrow{g}_{t,w_j} \tag{12}$$
$$\overrightarrow{o}_{t,w_j} = \sigma\left(\mathbf{W}_o\mathbf{X}_t + b_o\right) \tag{13}$$
$$\overrightarrow{h}_{w_j} = \overrightarrow{o}_{t,w_j}\tanh\left(\overrightarrow{c}_{t,w_j}\right) \tag{14}$$

and

$$\mathbf{W}_* = [w_{x*}, w_{h*}], \quad \mathbf{X}_t = \left[\text{vec}_{w_j}, h_{w_{j-1}}\right] \tag{15}$$

where w_{j-1} represents the feature at time $(t-1)$. Then, we use the elementwise sum to combine outputs of feature w_j from the two opposite LSTM layers to form its final semantic vector

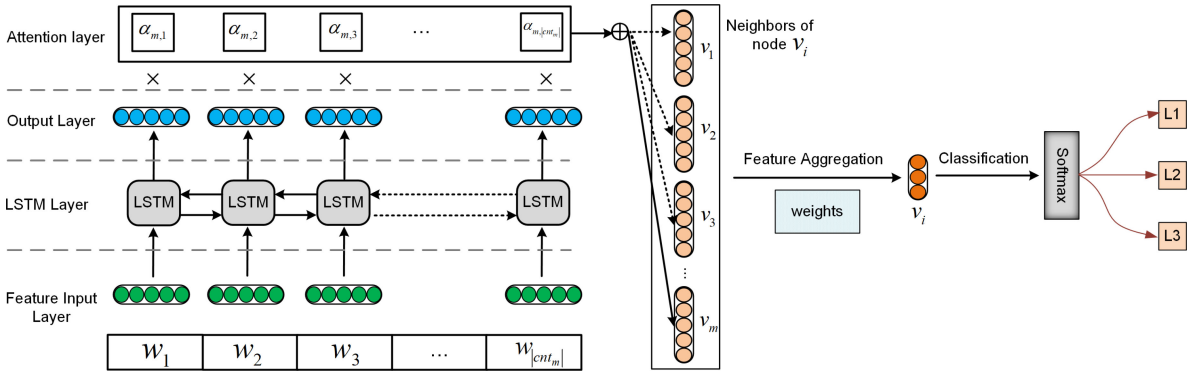


Fig. 4. Proposed FA-GCN model. Feature representations of nodes are dynamically learned by the LSTM network with attention mechanisms. Then, with the spectral-based convolutional filter, each node could gather the most important content features from itself and its neighbors to form the node representation. The feature and node representations are trained in a unified manner to optimize the collective classification objective by (28).

representation

$$h_{w_j} = \overrightarrow{h}_{w_j} \oplus \overleftarrow{h}_{w_j} \tag{16}$$

where $\overleftarrow{h}_{w_j} \in \mathbb{R}^{d_o}$ represents the vector output of w_j from the opposite temporal order, which is calculated in a similar way as (14).

B. Feature-Attention Mechanisms

Attention mechanisms have been widely used in many sequence-based tasks, such as sentiment classification [40] and machine translation [41], which are favorable designs allowing models to learn alignments between different modalities, that is, focusing on the most relevant neighborhood features that are helpful to the node classification task. In this article, we investigate two types of attention mechanisms both at the feature level.

Attention 1: Let the semantic vectors of all features in node content cnt_m be represented by $H_m = [h_{w_1}, h_{w_2}, \ldots, h_{w_{|\text{cnt}_m|}}]$, where $H_m \in \mathbb{R}^{|\text{cnt}_m| \times d_o}$ and $h_{w_{j=1,2,\ldots,|\text{cnt}_m|}}$ is calculated from (16). Inspired by a recent attention design that aims to capture the most important / relevant words in a given sentence for relation classification [42], we calculate a weight vector $\alpha_m = \{\alpha_j\}_{j=1,\ldots,|\text{cnt}_m|}$ for all features in H_m by

$$M = \tanh(H_m) \tag{17}$$

$$\alpha_m = \text{softmax}\left(MW_a^{\text{T}}\right) \tag{18}$$

where $\alpha_{m,j}$ is the attention weight for feature h_{w_j}. $M \in \mathbb{R}^{|\text{cnt}_m| \times d_o}$ is the nonlinear transformation of H_m, $W_a \in \mathbb{R}^{d_o}$ is a trained parameter vector shared across all nodes, and W_a^{T} is the transpose.

Attention 2: However, the above way of obtaining attention scores adopts a simple self-transformation without considering the neighborhood relationships. Therefore, we propose to perform a context-aware attention calculation, which determines the importance of neighborhood features by taking the corresponding context node into consideration. In Fig. 3, assume all neighbors (each node also aggregates information from itself) of node v_i are represented by a collection $\mathbb{N}_i = \{v_m\}_{m=1,\ldots,|\mathbb{N}_i|}$ and they have a shared contextual semantic vector computed by the elementwise sum of all individual feature vectors for node content cnt_i

$$h_{\text{context}_i} = \sum_{j=1}^{|\text{cnt}_i|} H_{i,j} \tag{19}$$

where $H_i = [h_{w_1}, h_{w_2}, \ldots, h_{w_{|\text{cnt}_i|}}]$ are semantic vectors of all features of node v_i calculated based on (16). Then, the weight score α_j for each feature $h_{w_j} \in H_m$ of the neighborhood node v_m is computed by using a bilinear term

$$\alpha_{m,j} = \text{softmax}\left(h_{w_j} W_b h_{\text{context}_i}^{\text{T}}\right) \tag{20}$$

where $W_b \in \mathbb{R}^{d_o \times d_o}$ is a trained parameter matrix and $h_{\text{context}_i}^{\text{T}}$ is a transpose.

By incorporating the attention weights, the final feature representation $X_m \in \mathbb{R}^{d_o}$ for neighborhood node v_m is formed by a weighted sum of all corresponding individual features by

$$X_m = \sum_{j=1}^{|\text{cnt}_m|} \alpha_{m,j} H_{m,j}. \tag{21}$$

C. Node Representation Learning

The spectral-based convolutional filter [13] is chosen as a key building component in our framework to gather features dynamically learned by the LSTM network for node representation learning, where each node aggregates features from itself and all its first-order neighbors at each convolution layer. In this article, we adopt a two-layer convolutional node representation learning process, where the embedding $X_i^{(1)} \in \mathbb{R}^{d_h}$ for each node v_i in the first layer is computed by

$$X_i^{(1)} = \sum_{m \in \mathbb{N}_i} W_0 X_m \tag{22}$$

where $W_0 \in \mathbb{R}^{d_h \times d_o}$ denotes the weight matrix and d_h is the dimension of node embeddings in the first layer. Assume embeddings of all nodes output in the first layer are represented by $X^{(1)} = \{X_i^{(1)}\}_{i=1,\ldots,n}$, and then the node embeddings in the second layer are computed by

$$O = \tilde{A}\text{ReLU}\left(X^{(1)}\right)W_1 \tag{23}$$

where $\tilde{A} = D^{-(1/2)}(I+A)D^{-(1/2)}$ is the normalized symmetric adjacency matrix with self-loops. $W_1 \in \mathbb{R}^{d_h \times l}$ is the parameter

Algorithm 1: FA-GCN for Noise Resilient Learning

Input : An information network: $G = (V, E, \mathbb{C})$
Output: The node embeddings: $O \in \mathbb{R}^{n \times l}$
Initialization: $i = 0$, training epochs I and labeled nodes $\mathcal{Y}_L$

while $i \leq I$ **do**
  for *a vertex* $v_m \in V$ **do**
    $H_m \leftarrow$ Learn the dense semantic vector for each feature in cnt_m by Eq. (16);
    $X_m \leftarrow$ Learn original node feature representation by Eq. (21).
  end
  $O \in \mathbb{R}^{n \times l} \leftarrow$ Learn node embeddings by Eq. (23);
  $\mathcal{L} \leftarrow$ Calculate classification loss by Eq. (28);
  $\left[W_*, W_b, W_0, W_1\right] \leftarrow$ Update feature representation learning weights W_* in Eq. (15), attention weight W_b in Eq. (20), and node embedding learning weights W_0 & W_1 in Eqs. (22) and (23);
  $i = i + 1$.
end

matrix that transforms each node embedding to a l-length vector. Finally, the output node embeddings $O \in \mathbb{R}^{n \times l}$ of the last layer are subsequently through a *softmax* classifier to perform a multiclass classification task by

$$Z = \text{softmax}(O) = \frac{\exp(O)}{\sum_i \exp(O_i)}. \tag{24}$$

Let Y be the one-hot label indicator matrix of all nodes, and the classification loss can be defined as the cross-entropy error by

$$\mathcal{L} = -\sum_{d \in \mathcal{Y}_L} \sum_{f=1}^{l} Y_{df} \ln Z_{df} \tag{25}$$

where $\mathcal{Y}_L$ is the set of node indices that have labels.

The weight parameters for feature representation learning (e.g., $w_{x*} = \{w_{xf}, w_{xi}, x_{xg}, w_{xo}\}$ and $w_{h*} = \{w_{hf}, w_{hi}, x_{hg}, w_{ho}\}$) and node representation learning (e.g., W_0 and W_1) are trained collectively using the gradient descent algorithm as in [3] and [13]. Since the feature representation learning of one node can be influenced by that of other neighborhood nodes, both the LSTM network parameters and GCN network parameters could vary dramatically without regularization, which might leads to the overfitting and instability problems. To mitigate these issues, we add an L2-norm regulation term on the loss function by

$$R_f^{\{2\}} = \sum_{w_x \in w_{x*}} \|w_x\|_F^2 + \sum_{w_h \in w_{h*}} \|w_h\|_F^2 \tag{26}$$

$$R_n^{\{2\}} = \|W_0\|_F^2 + \|W_1\|_F^2 \tag{27}$$

$$\mathcal{L} = -\sum_{d \in \mathcal{Y}_L} \sum_{f=1}^{l} Y_{df} \ln Z_{df} + \lambda_1 R_f^{\{2\}} + \lambda_2 R_n^{\{2\}} \tag{28}$$

where λ_1 and λ_2 are penalty terms to control the weight magnitude of the regularization terms on feature and node representation learning weight parameters, respectively. The collective learning process of FA-GCN is summarized in Algorithm 1.

TABLE I
BENCHMARK NETWORK CHARACTERISTICS

Items	Citeseer	Cora	DBLP
# Nodes	3,312	2,211	17,725
# Edges	4,732	5,001	52,890
# unique words	3,703	9,679	6,974
# average words per node	32	87	7
# Categories	6	7	4

V. EXPERIMENTS

A. Benchmark Data

Three widely used benchmark datasets are chosen in our experiments. Table I summarizes their network characteristics, and their domain information is described as follows.

The *Citeseer* dataset contains 3312 literature from six categories and 4732 links among them. Each publication is described by a text with average number of words of 32, where the word features in each node content are not ordered in a meaningful sequence (e.g., alphabetical order). There are 3703 unique words in the vocabulary.

The *Cora* dataset contains 2708 research papers from seven machine learning directions, such as *reinforcement learning* and *genetic algorithms*. Each paper corresponds to a category label. There are 5214 citation relations among these papers. Each paper is described by an abstract in the form of word sequence. There are 14 694 unique words in the vocabulary and the average number of words for each node is 90.

The *DBLP* dataset contains 10 310 publications from four research areas in computer science, including *database*, *data mining*, *artificial intelligence*, and *computer vision*. There are 52 890 edges in total and each publication is associated with a title in the form of word sequence. There are 15 135 unique words in the vocabulary and the average number of words for each publication is 8.

The detailed statistic information is summarized in Table I. We can observe that node contents in these networks are rather sparse. As we discussed previously, feature sparsity combined with feature noise would severely challenge most existing methods and degrade their embedding learning performance. In comparison, we propose to model feature correlations across all node contents, which can help to learn enriched semantics and mitigate the sparsity problems for some networks. More importantly, the attention mechanisms introduced in the proposed approach will help to identify important features for node interactions, thus reducing the influence of irrelevant or noisy features for improved embedding learning.

In the experiment, node contents on Cora and DBLP are presented as sequential word features, and it is straightforward to consider the feature dependencies or correlations for more accurate relationship modeling between nodes. In addition, to further evaluate the capacity of the proposed approach to model feature correlation and feature attention in a more general setting (e.g., sparse and noisy node content with disordering the original feature sequence), we also use the Citeseer dataset in which features are disordered, that is, features of all nodes are sorted by the alphabetical order. As we adopt a bidirectional LSTM network to learn the feature representations,

each feature is able to reach and interact with others within the same node content.

B. Baselines

We choose the following state-of-the-art comparative methods classified into three categories as follows.

Structure Only:

1) *DeepWalk* [8] preserves only the neighborhood relations between nodes by the truncated random walk, and uses the SkipGram model to learn the node embeddings.
2) *Node2vec* [21] adopts a more flexible neighborhood sampling process than DeepWalk, that is, biased random walk, to better capture the local structure (the second-order node proximity) and the global structure (the high-order node proximity).

Both Structure and Content Without Attention:

1) *TriDNR* [10] is a method that exploits network structure, node content, and label information for node representation learning. It is based on the assumption that network structures and contents can enhance each other to collectively determine the affinities between nodes.
2) *GCN* [13] is a state-of-the-art method that can efficiently model node relations from network structures and contents, where each node generates representation by adopting a spectral-based convolutional filter to recursively aggregate information from all its neighbors.
3) *FA-GCN*$_{cor}$ is a variant of our proposed method that models the feature correlations by a bidirectional LSTM network and then learns node representations based on the graph convolutional filter as GCN. The attention mechanism is not incorporated in this model.

Both Structure and Content With Attention:

1) *GAT* [18] is a method built on the GCN model. It introduces an attention mechanism at the node level, which allows each node specifies different weights to different nodes in a neighborhood.
2) *FA-GCN*$_{self}$ is a variant of our proposed method that learns feature representation based on the bidirectional LSTM network with introducing a self-transformation attention mechanism (attention 1 described in Section IV-B). It then learns node representations based on the graph convolutional filter.
3) *FA-GCN* is our proposed method that learns feature representation based on the bidirectional LSTM network and learns node representations based on the graph convolutional filter. The context-aware attention mechanism (attention 2 proposed in Section IV-B) is adopted in this method.

C. Experimental Settings

Node Classification: We first perform the supervised node classification based on the learned node representations, which is a widely used way to demonstrate the graph learning performance [8], [10], [13]. This experiment is conducted on the original networks, which allows us to compare different baseline methods for learning real-world graphs with sparse node contents (e.g., node features are very sparse in the three original networks). $p\%$ labeled nodes are randomly selected for training the model (e.g., classifier), which is then used to predict labels for the rest of nodes. The remaining nodes are split into two sets, where 10% (validation) for tuning the parameters and 90% (test) for testing the model performance. Similar to the literature [13], [18], we adopt *Accuracy* to measure the classification performance, where experiments are repeated five times w.r.t different portions of training data and the average performance and standard deviation are finally reported.

Noise Intervention: We further test the performance of the proposed models to handle artificial sparse and noisy content networks against all baselines. Two different types of noise interventions to the original node contents are examined.

1) *Type-I Noise:* We inject new words (not appeared in the original node content) randomly sampled from the entire corpus into each node content. The injection ratios (the number of new words over the number of original words in the node content) are ranging from 0.1 to 1.0.
2) *Type-II Noise:* We replace different ratios (e.g., between 0.05 and 0.5) of original words in the node content with new words randomly sampled from the entire corpus.

It is necessary to mention that the Type-I noise intervention will make each node content contain more irrelevant word features, while the Type-II noise intervention will make each node content become more erroneous and meanwhile, become more sparser (e.g., the original correct content features are removed). For both interventions, noise are introduced in a completely random manner, and we only alter the noise level to validate the algorithm performance. Since the contents for DBLP network are already very sparse (e.g., seven words per node), we only present the impacts of the Type-II noise intervention w.r.t Citeseer and Cora datasets.

In addition to noisy node features, noisy node structures are also harmful to most existing network embedding methods. Therefore, as an extended experiment, we are interested to investigate the performance of various methods when noisy structures are presented in the networks, although none of the baseline methods are designed to handle noise structures. We define the structural noisy intervention as Type-III Noise.

3) *Type-III Noise:* We introduce random noisy links (not appeared in the original network) into the network. The ratios (the number of new links over the total number of edges in the original network) of noisy links added between nodes range from 0.1 to 1.0.

Parameter Setting: Extensive experiments are designed to test the sensitivities of various parameters. We test the input and output feature representation dimensions d_i and d_o in the LSTM network between 20 and 200, and the training ratio $p\%$ of the labeled nodes between 0.1 and 0.5. For comparison, in this article, the default settings for d_i, d_o, and $p\%$ are 80, 80, and 0.4, respectively. d_h for Citeseer, Cora, and DBLP is set as 6, 7, and 4, respectively. Both the LSTM and GCN networks use the dropout technique to reduce the effect of overfitting, where dropout probabilities for LSTM and GCN are 0.2 and 0.3, respectively. The L2 norm regularization weight decay parameters λ_1 and λ_2 are, respectively, set as 5e-4 and 5e-4 for DBLP, and 5e-3 and 5e-4 for other datasets. We use the

TABLE II
NODE CLASSIFICATION RESULTS ON CITESEER (p% DENOTES PERCENTAGE OF LABELED NODES)

p%		10%	15%	20%	25%	30%	35%	40%	45%	50%
Methods	DeepWalk	28.34±1.01	30.17±0.86	31.04±0.95	32.17±0.80	32.73±1.03	33.88±0.79	34.13±0.92	34.58±0.96	35.08±0.64
	Node2vec	47.98±1.75	51.16±1.21	52.66±1.12	53.75±0.85	54.58±0.84	55.68±0.83	56.24±0.75	56.81±1.21	56.58±0.96
	TriDNR	49.45±0.84	51.95±0.29	53.40±0.92	55.31±0.49	55.20±0.57	55.18±0.84	56.79±0.57	57.14±1.01	57.13±0.56
	GCN	**71.06±0.47**	71.43±0.55	72.25±0.45	73.88±0.70	74.15±0.64	74.87±0.53	76.57±0.55	76.82±0.46	77.45±0.65
	GAT	*70.42±0.52*	**72.71±0.47**	*73.05±1.02*	74.36±0.70	74.65±0.77	76.10±0.46	77.15±0.38	77.60±0.51	78.47±0.64
	FA-GCN$_{cor}$	67.96±0.57	71.91±0.56	72.36±0.49	74.70±0.30	75.38±0.55	76.64±0.63	79.32±0.69	80.83±0.47	81.06±0.57
	FA-GCN$_{self}$	68.23±0.76	*72.07±0.53*	72.88±0.52	*75.04±0.43*	*75.45±0.82*	*77.06±0.62*	*79.60±0.52*	*81.29±0.55*	*81.66±0.64*
	FA-GCN	67.89±0.48	71.47±0.59	**73.09±0.52**	**75.38±0.60**	**76.89±0.60**	**77.50±0.41**	**80.39±0.60**	**81.43±0.58**	**82.38±0.46**

TABLE III
NODE CLASSIFICATION RESULTS ON CORA (p% DENOTES PERCENTAGE OF LABELED NODES)

p%		10%	15%	20%	25%	30%	35%	40%	45%	50%
Methods	DeepWalk	43.42±1.53	45.74±0.96	48.95±0.96	50.22±1.02	51.18±0.97	52.23±1.40	53.23±1.10	53.34±1.01	54.40±1.35
	Node2vec	68.66±1.83	71.19±1.78	73.51±1.50	74.79±1.35	76.36±1.01	77.10±1.05	78.03±0.95	78.43±1.33	78.78±1.12
	TriDNR	74.32±1.53	75.31±0.96	75.67±0.79	75.19±0.64	76.01±1.21	77.83±0.79	78.45±0.96	78.36±0.96	79.20±0.79
	GCN	*82.13±0.48*	*82.68±0.52*	84.14±0.80	84.68±0.41	85.42±0.65	85.40±0.40	87.34±0.35	85.25±0.65	87.11±0.65
	GAT	**82.60±0.47**	**83.34±0.64**	*84.33±0.48*	*85.15±0.39*	86.23±0.95	*86.73±0.72*	*87.62±0.41*	87.44±0.81	87.34±0.50
	FA-GCN$_{cor}$	79.82±0.57	80.95±0.42	83.33±0.53	84.29±0.44	84.60±0.33	85.68±0.54	86.77±0.57	87.07±0.39	87.34±0.71
	FA-GCN$_{self}$	79.20±0.48	82.21±0.54	84.24±0.37	84.84±0.61	*86.33±0.53*	86.45±0.60	87.32±0.45	**87.82**±0.55	*87.71±0.77*
	FA-GCN	78.20±0.42	82.49±0.46	**84.50±0.72**	**85.21±0.38**	**86.96±0.62**	**87.56±0.38**	**87.75±0.48**	*87.73±0.51*	**88.16**±0.61

TABLE IV
CLASSIFICATION RESULTS ON DBLP (p% DENOTES PERCENTAGE OF LABELED NODES)

p%		10%	15%	20%	25%	30%	35%	40%	45%	50%
Methods	DeepWalk	43.12±0.43	43.91±0.41	44.42±0.45	44.73±0.36	44.98±0.46	45.17±0.31	45.45±0.36	45.41±0.32	45.57±0.37
	Node2vec	76.08±0.39	76.99±0.29	77.40±0.28	77.56±0.27	77.75±0.36	77.92±0.25	77.96±0.36	78.02±0.26	78.16±0.30
	TriDNR	49.45±0.84	51.95±0.29	53.40±0.92	55.31±0.49	55.20±0.57	55.18±0.84	56.79±0.57	57.14±1.01	57.13±0.56
	GCN	**82.89±0.54**	**83.43±0.59**	83.66±0.48	83.76±0.68	84.31±0.85	84.29±0.40	85.18±0.56	85.40±0.43	**88.39±0.88**
	GAT	*82.07±0.39*	82.71±0.50	82.91±0.37	83.36±0.47	84.42±0.64	*85.72*±0.59	*86.15±0.45*	86.70±0.68	87.42±0.60
	FA-GCN$_{cor}$	79.85±0.67	82.16±0.83	83.62±0.51	83.84±0.48	84.51±0.38	85.01±0.58	85.87±0.43	86.50±0.48	87.76±0.66
	FA-GCN$_{self}$	80.95±0.50	82.45±0.57	*83.87±0.40*	*84.21±0.70*	*85.15±0.46*	85.16±0.68	85.85±0.50	*86.93±0.57*	87.53±0.61
	FA-GCN	80.50±0.44	*82.87*±0.61	**83.95±0.48**	**84.51±0.43**	**85.75±0.51**	**86.49±0.57**	**86.82±0.66**	**87.63±0.53**	*88.24±0.59*

Adam optimizer to train the model, where the learning rate and training epoch are set as 2e-3 and 200, respectively.

D. Experimental Results

1) Node Classification Performance: We first compare different baseline methods for addressing original graphs with sparse node features. The comparison w.r.t both sparse and noisy features (e.g., noise intervention performance) will be presented in the following part. Tables II–IV show the classification accuracy of all baselines on the three datasets, where the top three best results are bold faced, italic formatted, and underscored, respectively. From the results, we have the following four main observations.

From Tables II and III, we can conclude that methods incorporating both network structures and contents perform generally better than methods preserving only structures. For example, after modeling the text content of the Cora network, the average performance of TriDNR improved 52.5% over DeepWalk and 2.0% over Node2vec, respectively. The reason is probably because of the fact that both Citeseer and Cora are sparse networks, where structures fail to capture the holistic relations between nodes. In such case, network contents may can be leveraged to enhance node relationship modeling. The above phenomenon can be strengthen by the results from Table IV showing Node2vec performs significantly better than TriDNR on DBLP, where the DBLP network has denser connectivity but sparser content compared with Citeseer and Cora networks. On the other hand, the shallow models, such as DeepWalk, Node2vec, and TriDNR, suffer from the limitations of modeling complex node relations [5], that is, they all use random walk over the network to capture node relationships, but the random walk technique cannot differentiate the affinities of node neighborhood relations of varying hops. In comparison, other GCN-based methods (e.g., GCN, GAT, and FA-GCN$_{\text{cor}}$) can enforce a rigid neighborhood relations modeling by the efficient spectral-based convolutional feature aggregation process, where the learned node representations can naturally and precisely preserve the network structures and contents.

Based on the classification results over all three datasets, in most cases FA-GCN$_{\text{cor}}$ outperforms GCN, that is, an improvement of 1.7% w.r.t the Citeseer dataset. The superiority was actually brought by a more reasonable way of modeling the network content features in the proposed model. Existing GCN-based methods typically take the static content features as input, where features are treated as independent and nodes linking each others are assumed to have dependencies with their shared individual features. However, in many situations, (e.g., especially for text-described networks) each

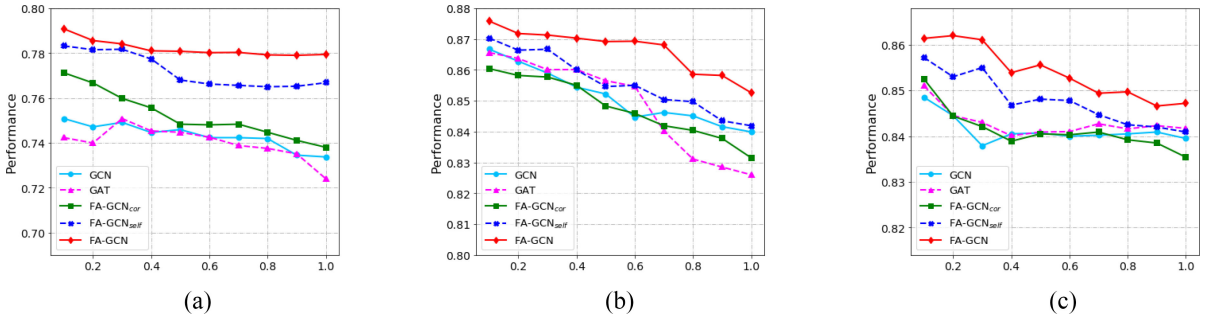


Fig. 5. Performance comparisons w.r.t. different levels of Type-I noise. The $x-$axis denotes noise levels, that is, 0.1 means adding 10% of random noise to each node (e.g., a node with ten words would be injected one random word as noisy node content). (a) Citeseer. (b) Cora. (c) DBLP.

feature (e.g., word) usually not only appears to represent a single meaning but also has correlations with others to reveal a complete and complex semantic. In comparison, the proposed approach adopts a bidirectional LSTM network to effectively model the feature correlations for more accurate node relations modeling, especially when the node features are too sparse to capture accurate relations between nodes. The performance gain has demonstrated the benefits of the proposed model to learn accurate feature semantics from sparse node features. However, it is interesting to note that FA-GCN and its variants (FA-GCN$_{\text{cor}}$ and FA-GCN$_{\text{self}}$) are generally inferior to GCN and GAT when the percentages of labeled nodes are relatively small (e.g., 10% and 15%). The possible reason is that FA-GCN and its variants need to train an LSTM component; thus, more labeled nodes are desired to train the parameters and produce an optimal model. This point is supported by that with the increased percentage of labeled nodes, FA-GCN starts to show superior performance compared with GCN and GAT.

As can be seen from Tables II–IV, models (e.g., GAT, FA-GCN$_{\text{self}}$, and FA-GCN) with incorporating either node-level attention mechanism or feature-level attention mechanism perform generally better than the basic GCN model. For example, on the Citeseer dataset, the average performance of GAT improved 0.9% over GCN, and FA-GCN$_{\text{self}}$ and FA-GCN improved 2.2% and 2.6%, respectively. In general, edges in a graph could reveal complex relationships between nodes, that is, in the citation network, a paper may cite many others of various subject matters, and in the social network, a user may connect many friends of different degrees of affinities. GCN enforces each node to indiscriminately aggregate information from all neighbors, which is inflexible and insufficient to model neighborhood relations between nodes. In comparison, GAT and the proposed attention model allow each node attends important neighborhood nodes or their features for differentiable neighborhood features aggregation, which is helpful to accurately model node relationships especially when node features may be contaminated with noises.

For attention-based methods, we can observe from Tables II and IV that FA-GCN$_{\text{self}}$ outperforms GAT in most cases (e.g., the average performance improved 1.3% w.r.t Citeseer dataset). The reason lies in that FA-GCN$_{self}$ adopts a more fine-grained attention mechanism at the feature level, against GAT at the node level. As we know, features in a node content could be irrelevant or noises, where two node sharing many identical features cannot guarantee they are highly similar. For example, in the citation network, the abstract of a publication has rich word features in which many are not distinguishing features (e.g., irrelevant or noisy features) to reveal the accurate topics involved. The node-level attention assumes that all features in each node content contribute equally, while the feature-level attention in FA-GCN$_{\text{self}}$ is able to assign higher weights to the most influential features for node interactions, thus reducing the contribution of noisy features. In addition, we can observe from all three result tables that FA-GCN can achieve even better performance than FA-GCN$_{\text{self}}$. This is because that FA-GCN has considered the contextual information while calculating each feature attention of all neighboring nodes, which enables each node have more capacity to select relevant features (from noisy node content) and meanwhile, learn accurate feature semantics (from sparse node content) aligned with the final node classification. The experimental results demonstrated effectiveness of the proposed attention models for extracting significant features for improved node interaction modeling.

2) Noise Intervention Performance: Fig. 5 reports the performance of various GCN-based methods on all three graphs w.r.t different ratios of Type-I noise. As can be seen, with the increase of noise level, performances of all GCN-based methods, such as GCN, GAT, and FA-GCN$_{\text{cor}}$, tend to decline to some degrees. The phenomenon is mainly caused by the fact that in the spectral-based graph convolution learning process, nodes are forced to aggregate content features from their respective neighbors while modeling node interactions. Once node contents are floated with noisy features, the content similarities between nodes would become less likely to reflect accurate neighborhood relationships, thus blurring interactions between nodes and causing performance degradation. Nevertheless, from Fig. 5(a), we can observe the proposed FA-GCN is less sensitive to noises. The reason is mainly twofold: 1) feature dependencies across all node contents have been captured in FA-GCN, which can help to alleviate impacts of feature sparsity and noise for some individual nodes and 2) the introduced feature-level attention models are helpful to select important features from the noisy content, which has guaranteed, to some extent, the accurate neighborhood relations modeling. In addition, Fig. 5(b) and (c) shows that FA-GCN$_{\text{self}}$ and FA-GCN both outperform other baselines in most cases, which again demonstrated that the

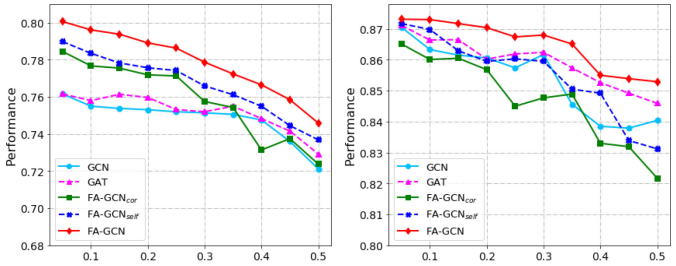


Fig. 6. Performance comparisons w.r.t. different levels of Type-II noise. The $x-$axis denotes noise levels, that is, 0.1 means replacing 10% of words in each node as noise (e.g., a node with ten words would have one word being replaced with a random word). (a) Citeseer. (b) Cora.

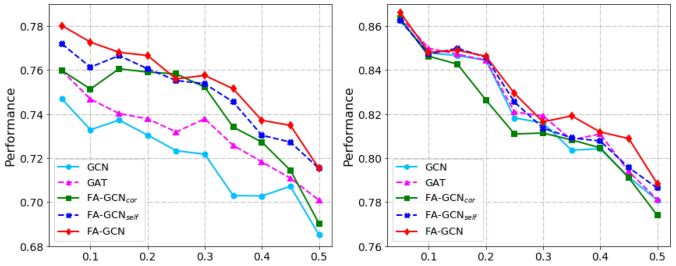


Fig. 7. Performance comparisons w.r.t. different levels of Type-III noise. The $x-$axis denotes levels of noisy links, that is, 0.1 means adding 10% of random noisy links to the network. (a) Citeseer. (b) Cora.

feature-level attention is beneficial to promote the GCN-based noise-resilient embedding learning.

Fig. 6 shows the impact of Type-II noise intervention. We can observe that it has a significant impact on the performance of all comparative methods, that is, with the noise ratio increased from 0.1 to 0.5, the performance of GCN decreased to 3.4% and 2.3% on Citeseer and Cora, respectively. The reason is that the Type-II noise intervention makes the node contents in the two studied networks Citeseer and Cora more sparser combined with noises, which severely deteriorates the node relations modeling for all compared methods. Nevertheless, the proposed FA-GCN model still performs better than other methods in most cases, which again demonstrated the effectiveness of the proposed models for sparse and noisy content network learning. The benefits of FA-GCN come from capturing feature correlations and learning feature importance to address sparse and noisy content features for improved embedding learning.

Interestingly, the results in Figs. 5 and 6 show that although all methods suffer from performance loss due to noise impact, none of the compared method becomes completely random, even though a significant amount of noise (e.g., 100% for Type-I noise and 50% for Type-II noise) are introduced to the data. We believe that this is mainly attributed to the fact that node relationships are largely constrained by the graph topology, which is considered not noisy in this study. Although node content are severely contaminated or missing, the model performance would not deteriorate to be completely random, due to the contribution of network topology used for learning. Meanwhile, although node content are noisy, node labels still provide useful information to support the learning. Evidently, a recent study [43] also shows that GCN-based methods can achieve rather good performance on graphs without using any of the node content.

Fig. 7 shows the performance of various methods when Type-III noise (noisy links) is introduced in the network. Compared to Type-I noise [Fig. 5(a) and (b)], we can observe from Fig. 7 that all methods degrade more significantly with the increase of noise levels. This is because the GNNs-based methods directly rely heavily on the topological structures to perform node relation modeling. FA-GCN is mainly designed to handle noisy features in the network, and it is lacking the ability to handle noisy structures like other baselines. We leave it as future work to further extend FA-GCN to be able to address the Type-III noise.

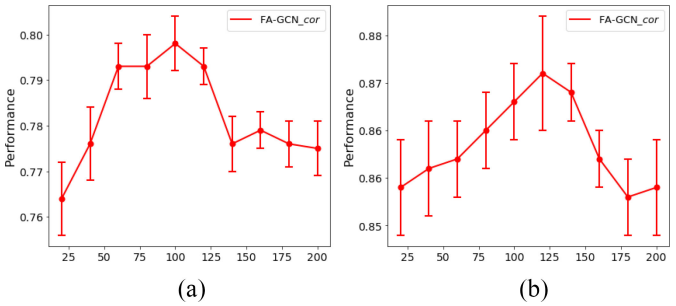


Fig. 8. Impact of the input feature vector dimension d_i. (a) Citeseer. (b) DBLP.

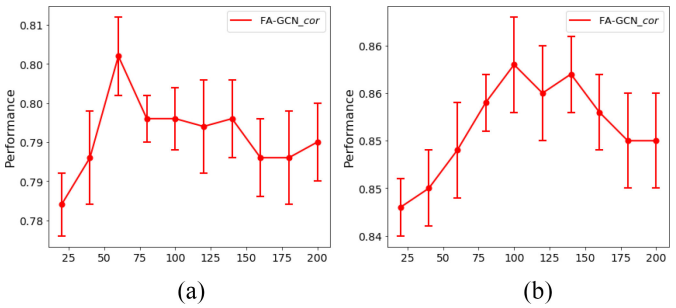


Fig. 9. Impact of the output feature vector dimension d_o. (a) Citeseer. (b) DBLP.

3) Parameter Influence: They are three hyperparameters, d_i, d_o, and d_h, that are important in the feature and node representation learning process. Extensive experiments are designed to test their sensitivities on Citeseer and DBLP datasets. d_i controls the dimension of the feature vector fed into the LSTM network. Fig. 8 shows on both datasets the performance changes in a limited range, which first increases and then drops with the increase of d_i. If d_i is set too small, the semantic representation ability of each word will be limited. In other words, smaller dimension may cause vector representations for different words less expressive and indistinguishable, thereby blurring the content discrepancy between different nodes. On the contrary, if d_i is set too large, the semantic information carried by each word would be overly represented and highly distributed, making the LSTM model

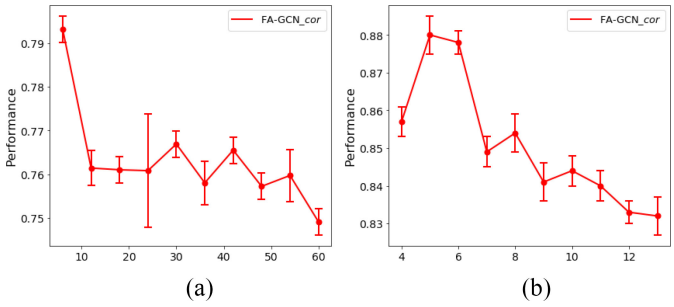


Fig. 10. Impact of the hidden node vector dimension d_h. (a) Citeseer. (b) DBLP.

ineffective to learn helpful information from the vector representation of each word, and meantime, making the entire training inefficient to optimize the high-dimensional model parameters.

d_o indicates the dimension of the feature representation output by the LSTM network. Analogical to d_i, if d_o is set too small or too large, it would make node content representation less expressive or make the GCN model ineffective to capture/aggregate highly distributed node content semantics in the convolution learning of relationships among nodes. As can be seen from Fig. 9, the performances first slightly increase and then decrease within a very small range after the dimensions are set as 60 and 100 for Citeseer and DBLP, respectively.

Fig. 10 shows the impact of parameter d_h, which represents the dimension of node representing out by the first convolutional layer in FA-GCN. Previous studies [44] show that d_h is an important parameter, which could significantly impact the GCN learning performance. We test impact of d_h on Citeseer between 6 and 60, and on DBLP between 4 and 13. The results show it has a significant impact on the performance, where the performance dramatically declines after 6 and 5 for Citeseer and DBLP, respectively.

VI. Conclusion and Future Work

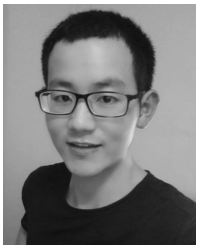
In this article, we studied noise resilient learning for networks with sparse and noisy node content. We argued that sparse, noisy, and erroneous graph content are ubiquitous. They present critical challenges to many graph learning methods that rely on network content to constrain and measure node relationships. To tackle feature sparsity, we first proposed to represent content features as dense vectors by an LSTM network, which leverages feature semantic correlation and dependency globally from all node contents to learn dense vector for each feature. After that, we introduced a feature attention mechanism that allows each node to vary feature weight values with respect to different neighbors, allowing our method to minimize the noise impact and emphasize on consistent features between connected nodes. As a result, each node can gather the most important content features from itself and its neighbors to learn its node representation. The effectiveness of the proposed models has been validated on three sparse content benchmark networks. The experiments on noise-free and noisy networks, including Type-I and Type-II noise interventions by either injecting noise into the node content or replacing correct content features with error ones, confirm that the proposed method outperforms state-of-the-art methods, such as GCN and GAT. Our method is less sensitive to erroneous graph contents, and is noise resilient for learning node representation compared with existing methods.

In this work, we are mainly focused on networks containing noisy node features. However, networked data may suffer from other types of noises, such as noisy links and even hybrid noises combining noisy node features and noisy links, which can severally challenge most existing network embedding methods. Building on the foundation of the proposed FA-GCN, future work is expected to design additional learning components (e.g., edge attention) that can minimize the impact of noisy links.

References

[1] H. NT, J. J. Choong, and T. Murata, "Learning graph neural networks with noisy labels," in *Proc. Int. Conf. Learn. Represent. (ICLR) 2nd Learn. Ltd. Labeled Data (LLD) Workshop*, 2019, pp. 1–5.
[2] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.
[3] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," 2018, *arXiv:1809.05679*.
[4] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. 32nd Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2018, pp. 5171–5181.
[5] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 3–28, Mar. 2020.
[6] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "*struc2vec*: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2017, pp. 385–394.
[7] T. M. V. Le and H. W. Lauw, "Probabilistic latent document network embedding," in *Proc. IEEE Int. Conf. Data Min.*, Shenzhen, China, 2014, pp. 270–279.
[8] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2014, pp. 701–710.
[9] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2111–2117.
[10] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proc. IJCAI*, 2016, pp. 1895–1901.
[11] J. Chang and D. Blei, "Relational topic models for document networks," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2009, pp. 81–88.
[12] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph and text jointly embedding," in *Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP)*, 2014, pp. 1591–1601.
[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[14] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.*, 2018, pp. 593–607.
[15] G. Qu, S. Hariri, and M. Yousif, "A new dependency and correlation analysis for features," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 9, pp. 1199–1207, Sep. 2005.
[16] Z. Kang, H. Pan, S. C. H. Hoi, and Z. Xu, "Robust graph learning from noisy data," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 1833–1843, May 2020.
[17] J. Pujara, E. Augustine, and L. Getoor, "Sparsity and noise: Where knowledge graph embeddings fall short," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, 2017, pp. 1751–1756.
[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019, *arXiv:1901.00596*.
[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.
[21] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2016, pp. 855–864.

[22] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
[23] C. Tu, H. Liu, Z. Liu, and M. Sun, "CANE: Context-aware network embedding for relation modeling," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist.*, 2017, pp. 1722–1731.
[24] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
[25] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2016, pp. 3844–3852.
[26] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.
[27] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 7444–7452.
[28] N. Manwani and P. S. Sastry, "Noise tolerance under risk minimization," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 1146–1151, Jun. 2013.
[29] B. Du, T. Xinyao, Z. Wang, L. Zhang, and D. Tao, "Robust graph-based semisupervised learning for noisy labeled data via maximum correntropy criterion," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1440–1453, Apr. 2019.
[30] B. Frénay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, May 2014.
[31] X. Lu, L. Ming, W. Liu, and H.-X. Li, "Probabilistic regularized extreme learning machine for robust modeling of noise data," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2368–2377, Aug. 2018.
[32] C. Zhang, Z. Yu, H. Fu, P. Zhu, L. Chen, and Q. Hu, "Hybrid noise-oriented multilabel learning," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2837–2850, Jun. 2020.
[33] Z. Qiu, W. Hu, J. Wu, Z. Tang, and X. Jia, "Noise-resilient similarity preserving network embedding for social networks," in *Proc. IJCAI*, 2019, pp. 3282–3288.
[34] S. Pan, J. Wu, X. Zhu, and C. Zhang, "Graph ensemble boosting for imbalanced noisy graph stream classification," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 954–968, May 2015.
[35] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 695–704.
[36] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2019, pp. 1399–1407.
[37] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2018, pp. 2847–2856.
[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[39] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," 2016, *arXiv:1601.06733*.
[40] M. Yang, W. Tu, J. Wang, F. Xu, and X. Chen, "Attention-based LSTM for target-dependent sentiment classification," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 5013–5014.
[41] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*.
[42] P. Zhou *et al.*, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist. Vol. 2 Short Papers*, vol. 2, 2016, pp. 207–212.
[43] H. Taguchi, X. Liu, and T. Murata, "Graph convolutional networks for graphs containing missing features," *Future Gener. Comput. Syst.*, vol. 117, pp. 155–168, Apr. 2021.
[44] L. Zhao *et al.*, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.



Min Shi (Student Member, IEEE) received the Ph.D. degree in computer science from Florida Atlantic University, Boca Raton, FL, USA, in 2020.

He is currently a Postdoctoral Research Fellow with the Department of Genetics, Washington University School of Medicine, St. Louis, MO, USA. His research interests include machine learning, data mining, social network, bioinformatics, and service computing.

Yufei Tang (Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Rhode Island, Kingston, RI, USA, in 2016.

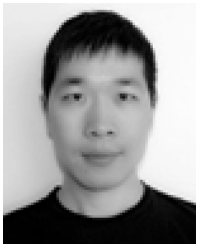
He is currently an Assistant Professor with the Department of Electrical Engineering and Computer Science and a Faculty Fellow with the Institute for Sensing and Embedded Network Systems Engineering, Florida Atlantic University, Boca Raton, FL, USA. His research interests include machine learning, data mining, dynamical systems, and renewable energy.

Dr. Tang was a recipient of the IEEE International Conference on Communications Best Paper Award in 2014, the National Academies Gulf Research Program Early-Career Research Fellowship in 2019, and the National Science Foundation CAREER Award in 2022.

Xingquan Zhu (Senior Member, IEEE) received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2001.

He is currently a Professor with the Department of Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL, USA. His current research interests include data mining, machine learning, multimedia computing, and bioinformatics. Since 2000, he has been authored or coauthored over 270 refereed journal and conference papers in these areas, including three Best Paper Awards and three Best Student Paper Award.

Prof. Zhu was an Associate Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING from 2008 to 2012 and since 2014. Since 2017, he has been an Associate Editor of the *ACM Transactions on Knowledge Discovery From Data*.



Yuan Zhuang (Member, IEEE) received the Ph.D. degree in geomatics engineering from the University of Calgary, Calgary, AB, Canada, in 2015.

He is a Professor with the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan, China. He has coauthored over 80 academic papers and 18 patents. His current research interests include multi-sensors integration, real-time location systems, wireless positioning, the Internet of Things (IoT), and machine learning for navigation applications.

Prof. Zhuang has received over ten academic awards. He is an Editorial Board Member of *Satellite Navigation* and IEEE ACCESS, a Guest Editor of the IEEE INTERNET OF THINGS JOURNAL, and a Reviewer of over 15 IEEE journals.

Maohua Lin received the B.Eng. degree from Fuzhou University, Fuzhou, China, in 2007, the M.Eng. degree from Nanchang University, Nanchang, China, in 2013, and the Ph.D. degree in mechanical engineering from Florida Atlantic University, Boca Raton, FL, USA, in 2019.

He is currently a Research Scientist with Florida Atlantic University. His research interests include computational model, machine learning, sensor design, and medical device.

Jianxun Liu received the M.S. degree in computer science from the Central South University of Technology, Changsha, China, in 1997, and the Ph.D. degree in computer science from the Shanghai Jiao Tong University, Shanghai, China, in 2003.

He is currently a Professor and the Dean with the School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. His research interests include service computing and cloud computing and big data.