This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3085882, IEEE Internet of Things Journal

Core-GAE: Towards Generation of IoT Networks

Qi Luo, *Student Member, IEEE,* Dongxiao Yu, *Senior Member, IEEE,* Yanwei Zheng, *Member, IEEE,* Hao Sheng, *Member, IEEE,* and Xiuzheng Cheng, *Fellow, IEEE*

***Abstract*—To realize simulation experiments in large-scale Internet of Things (IoT) networks, this work studies the utilization of deep graph generative models to generate IoT networks, which can provide an economic approach facilitating IoT to meet the requirements of real-time performance, interoperability, energy efficiency, and coexistence. In IoT, nodes have different attributes, different connection ways with surrounding nodes, and different compactness of the region, which pose great challenges for network generation. By leveraging the properties of k-core and variational autoencoder during network generation, we propose a variable graph autoencoder called Core-GAE incorporating coreness of nodes. In contrast to previous graph generative models, Core-GAE can preserve the local proximity similarity and maintain the global structural features simultaneously when learning the structural features of graphs. All three of the tasks we experimented with on four datasets show that Core-GAE exhibits better performance than previous ones.**

***Index Terms*—Iot Network, Deep Generative Model, Graph Autoencoder, Core Decomposition.**

I. Introduction

The Internet of Things (IoT), as a basic pillar of digital manufacturing, achieves the ubiquitous connection between objects and devices through various possible network access, and intelligently senses, identifies and manages objects [1]. Productivity and efficiency can be significantly improved through IoT for many important applications, such as public utility companies [2], agricultural producers [3] and healthcare providers [4]. Internet of Things is composed by ubiquitous devices. The purpose of the IoT is to make the environment and the IoT converge by perceiving the surrounding environment, transmitting and processing the acquired data, and then feeding it back into the environment. More and more devices are accessing the IoT networks. It is estimated that 28 billion devices will be connected to the IoT networks in 2021 [5]. As a result, the volume of IoT devices can provide analytical solutions and lead to better production practices.

Enormous IoT networks are emerging with the IoT development, which simultaneously brings some challenges [6], [7], such as the requirements for energy-efficient operation of the network, the coexistence of diverse devices and the need for real-time data feedback. In order to deal with all kinds of problems that may appear in large-scale IoT networks, it should first carry out verification experiments in a simulated environment. In other words, through the generative model, a large-scale IoT network is generated, and specific simulation experiments can be completed. By generating IoT networks, we can predict the emergency situation and avoid unnecessary losses. Fig. 1 shows an example of IoT network generation.

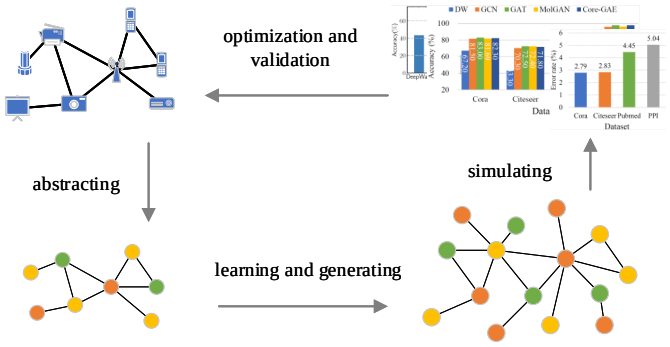


Fig. 1. An example of IoT network generation.

IoT networks can be abstracted as graphs, where the devices in the network can be regarded as the nodes, and the communication between devices can be regarded as the edges or links in the graph. Therefore, to generate a large-scale IoT network, we only need to abstract the devices and connections in the network as elements in the graph, and design a learning model to learn the feature of the network, including the topology structural information, features of nodes and connections. Then, through the learned model, a larger graph (ie, a larger IoT network) can be generated and has similar attributes and structures to the previously learned network.

Benefiting from rapid development of deep learning, the deep generation models have achieved great success in the areas of image [8], speech [9] and natural language [10]. Currently, the main applications are word prediction [11], time series generation [12], music improvisation [13], image generation [14] and video synthesis [15]. Therefore, a natural question is whether deep generative models can be utilized in generating graph structures. However, this task faces great challenges. On the one hand, the connection of the nodes in graphs is arbitrary, and there is no clear way to construct the graph in a linear manner according to a series of steps [16]. On the other hand, the learning process of graph representation is challenging. In the training process without stepwise supervision, the iterative construction process of the discrete structure of the graph involves discrete decisions, which are not differentiable, so it is a problem for the training process of backpropagation [17].

Despite of the great difficulities due to the discrete structure

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102600, in part by NSFC under Grant 61971269, and Grant 61832012, in part by Shandong University multidisciplinary research and innovation team of young scholars under Grant 2020QNQT017.

Q. Luo, D. Yu , Y. Zheng (Corresponding author), X. Cheng are with the School of Computer Science and Technology, Shandong University, Qingdao, P.R. China. Email:luoqi2018@mail.sdu.edu.cn, {dxyu, xzcheng}@sdu.edu.cn.

H. Sheng is with State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing, P.R. China. Email:shenghao@buaa.edu.cn.

of graph, the development of Graph Convolution Networks (GCN) [18] shed some light on deep graph generation. There have been some results on deep graph generation [19]–[24] presented based on GCN. But there are not existing results on IoT network generation, becauese of the special features of IoT, e.g., nodes with different attributes, different connection types with surrounding nodes, and different compactness of the region [25].

To address the aforementioned issues, we study the graph autoencoder model for IoT generation, considering the complex graphs abstracted from IoT. We propose a k-core [26] based deep graph generator, called Core-GAE, to learn graph data distribution. In contrast to the previous graph generative models, Core-GAE can preserve the local proximity similarity and maintain the global structural features simultaneously. Core-GAE is decomposed into two components, encoder and decoder. The encoder is responsible for representing the original graph structure as low dimensional vectors, and the decoder is responsible for reconstructing the graph with low dimensional vectors. This decoupling operation makes the model more flexible and enables it to learn structural features of graphs. Fig. 2 illustrates the framework of Core-GAE. Experimental results on real-world graphs show that Core-GAE preforms better on many tasks than previous methods.

The rest of our paper consists of the following sections. Section II briefly reviews existing graph generative methods including probabilistic graph models, sequential models and global models. Section III summarizes notations and definitions used in this paper and formulates the problem of deep graph generation. Section IV introduces our coreness based graph autoencoder model. The experiment results and analysis are illustrated in Section V. The conclusion and future work are given in Section VI.

II. Related work

In this section, we discuss the existing works on graph generation.

The earliest random graph model of graph generation was developed by Erdős & Rényi [27]. The Erdős & Rényi model assumed an independent identical probability for each possible edge, which leads to rich mathematical theory on random graphs. But this model is too simplistic to model more complex graphs, and the independent and identically distributed assumption violates real-world graphs.

Barabási & Albert [28] involved *preferential attachment* to generate graphs. It is assumed that the more neighbors a node has, the more likely it will be linked to new nodes added to the graph. Such models typically capture only one attribute of the graph and are not flexible enough to model a wide range of graphs. Some small-world models such as Watts & Strogatz [29] aim to capture the small diameter and local clustering coefficient properties in graphs. $p*$ model [30] is an exponential random graph model which estimates networks from the field of social analysis. However, $p*$ model often focuses on the local structural features of the graphs. Leskovec et al. [31] proposed the Kronecker graph model which is capable of modeling multiple properties of graphs, but it still has limited capabilities for easy-to-handle mathematical analysis.

In recent years, the deep graph generative model has achieved great success in the field of graph representation learning and graph generation. There are many applications of deep graph generation, ranging from the discovery of new molecular structure to the modeling of social networks and other fields have achieved good results. Graph autoencoders are deep neural architectures which learn graph representations in a latent low-dimensional feature space. Deep graph generative models aim at learning the distribution representations of graphs by encoding graphs into low-dimensional vectors and decoding the learned embeddings to reconstruct new graphs. Most graph generative models focus on molecular graph generation problems, which have many practical applications in drug discovery [32]. The deep graph generative models is mainly developed into two categories, which are sequential methods and global methods.

Sequential methods [19], [20], [33]–[35] generates new graphs by increasingly proposing nodes and edges step by step. For example, when generating molecular graphs, deep convolutional neural networks and recursive neural networks are adopted as encoder and decoder respectively to simulate the generation process of string representations for molecular graphs. Although these methods are applied in specific areas, they can also be applied to generation of general graphs. Nodes and edges can be added iteratively to the increasing graphs until specific conditions are met through these methods. However, due to the existence of generation periods and the idea of serializing graph information, the sequential methods may lose the topological structural information of graphs.

Global methods [21], [23], [24], [36], [37] output a graph all at once. VAE [36] is a latent variable-based model that pairs a generator with an inference network. The precondition of Graph Variational Autoencoder [21] is to set that the generation of nodes and edges is independent and unrelated, and its existence is treated as an independent random variable. The model outputs the adjacency matrix, node attribute matrix, and edge attribute matrix of the resulting graph. However, how to effectively preserve the global structural information of the reconstructed graph remains a challenge.

Regularized Graph Variational Autoencoder [22] further improves Graph Variational Autoencoder by imposing validity constraints to regularize the output distribution of the decoder. Some methods are inspired by Generative Adversarial Networks [38], [39], and generative adversarial models for graph generation are proposed. Molecular Generative Adversarial Network [23] combines convGNN [40], GAN [41] and reinforcement learning, which consists of a generator and a discriminator and is able to generate molecules graphs with specific required attributes. NetGAN [24] combines LSTM [42] and Wasserstein GAN [43] to generate graphs based on random walks. NetGAN uses reliable random walk data to train generator in LSTM network, and uses discriminator to identify false random walk from actual random walk to confront generator. Although not specially trained, the model can generate graphs and show network patterns without specifying them explicitly in the model definitions.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3085882, IEEE Internet of Things Journal

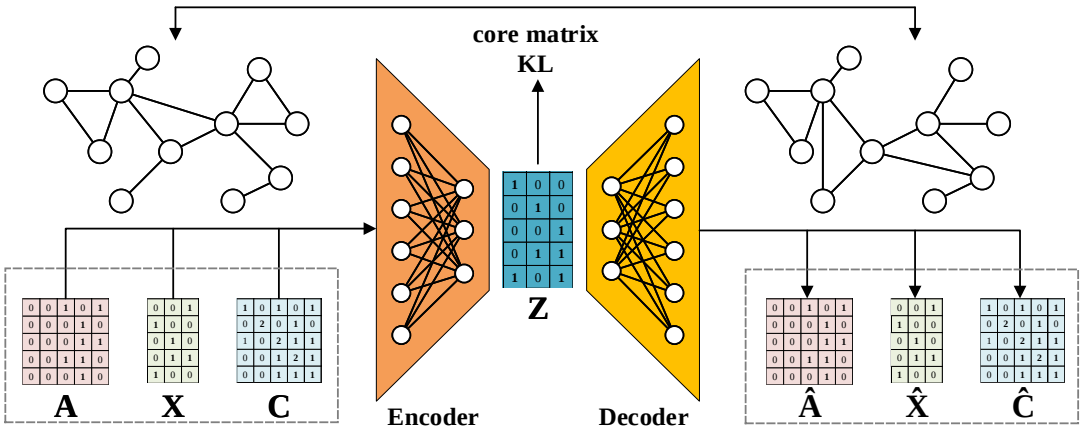


Fig. 2. Illustration of the core based variational graph autoencoder. We let the adjacency matrix, feature matrix, and core matrix be input of the encoder. The graph encoder embeds the graph into continuous representation $\mathbf{Z}$. Then the decoder output the reconstruct graph $\hat{G}$. Reconstruction ability of the autoencoder is expedited by approximating G and $\hat{G}$.

III. Preliminaries

We consider an undirected and unweighted graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V}^2$ is the set of edges, $N = |\mathcal{V}|$. The adjacency matrix of G is $\mathbf{A}$, with elements

$$A_{ij} = \begin{cases} 1 & ,(v_i, v_j) \in \mathcal{E}, \\ 0 & ,otherwise. \end{cases} \tag{1}$$

The degree matrix of G is a diagonal matrix, denoted as $\mathbf{D}$, with elements

$$D_{ij} = \begin{cases} \sum_{j=1}^{N} A_{ij} & ,i = j, \\ 0 & ,otherwise. \end{cases} \tag{2}$$

Definition 1 (k-core): Given a graph G, a k-core is a maximal connected subgraph where the degree of each node is not less than k.

k-core satisfies hierarchical property, a k-core must be contained in a $(k+1)$-core. The core number (coreness) of a node v is the k value of the maximum k-core in which v is belonging, denoted as $core(v)$.

The matrix $\mathbf{Z} \in \mathbb{R}^{N \times F}$ summarizes the embeddings of nodes, F is the dimension of learning embedding, $\mathbf{X}$ is a given feature matrix summarizing the node attributes.

The notations and descriptions are summarized in Table I.

TABLE I
THE SUMMARY OF NOTATIONS.

Notation	Description
G	an undirected, unweighted graph
$\mathcal{V}$	a set of nodes
$\mathcal{E}$	a set of edges
$\mathbf{A}$	the adjacency matrix of graph
$\mathbf{D}$	the degree matrix of graph
$\mathbf{X}$	the feature matrix of nodes
$\mathbf{C}$	the coreness matrix of nodes
$\mathbf{Z}$	the learning embedding matrix of nodes

Problem formulation. Given a graphs G, Deep Graph Autoencoder aims to learn samples from the data distribution by employing deep graph neural networks and generate new graphs. Specifically, this can be done by either learning the distribution of graph first and then sample from the estimated distribution to generate a new graph.

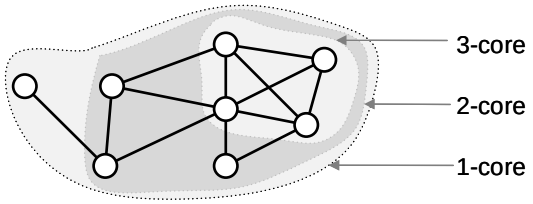


Fig. 3. An example of k-core.

IV. k-Core based Graph Autoencoder

We propose the k-core based graph autoencoder in this section. Firstly, we introduce our graph encoder and decoder, then we propose the loss function and the learning procedure. The encoder is defined by posterior $q_\phi(\mathbf{z}|G)$, and the decoder is defined by a generative distribution $p_\theta(\mathbf{z}|G)$, where ϕ and θ are learned parameters.

A. Graph Encoder

Firstly, we introduce our graph encoder $q(\mathbf{Z}|\mathbf{X}, \mathbf{A})$ based on graph convolutional networks. The general graph convolutional framework is divided into two components: node feature transformation and neighboring feature aggregation, which is defined as

$$\mathbf{Z} = h(f(\mathbf{X}), g(\mathbf{A})), \tag{3}$$

where f is the feature transformation function, g is the propagation rule, h is the feature aggregation function, $\mathbf{Z}$ is the learned embedding matrix. Through the input graph adjacency matrix, node feature matrix, and the defined propagation rules, the final output is the node representation matrix.

To preserve the abundant global structural information in embedding matrix, we use the coreness of nodes as the additional structural attribute feature. The coreness of a node

can effectively reflect the importance of the node in the graph. The coreness feature of each node can be represented by a sequence of coreness of its neighbors. In the following, we present the representation of coreness matrix. The coreness of all nodes can be computed in $O(|\mathcal{E}|)$ time [26]. Let $C = \{C_0, C_1, \cdots, C_{k_{max}}\}$ be the k-cores of graph G, where k_{max} is the max k value of k-core. The k-cores follows the property,

$$C_{k_{max}} \subseteq C_{k_{max}-1} \subseteq \cdots \subseteq C_1 \subseteq C_0 = G. \tag{4}$$

An example of k-core is shown in Fig. 3. According to the property, the k-core that an edge locates in is determined by its endpoint with the smaller coreness. Thus, the elements in coreness matrix $\mathbf{C}$ of G is

$$C_{ij} = \begin{cases} \min\{core(v_i), core(v_j)\} & ,(v_i, v_j) \in \mathcal{E}, \\ core(v_i) & ,i = j, \\ 0 & ,otherwise. \end{cases} \tag{5}$$

The relationship between each node's coreness and its neighbor's coreness can be reflected in this matrix. This matrix can also be regarded as the coreness of each edge, i.e., the edge feature matrix defined by coreness. For each training graph, we firstly compute the coreness matrix and prepare for input. We adopt the coreness matrix as the additional input of the graph convolutional network. Formally, our two layer core based graph convolutional network (CGCN) is defined as

$$CGCN(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}} ReLU(\tilde{\mathbf{A}}\mathbf{C}\mathbf{X}\mathbf{W}_0)\mathbf{W}_1, \tag{6}$$

where $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix, $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$, $\mathbf{I}$ is unit diagonal matrix, $\mathbf{W}$ is a learnable weight matrix in the graph convolutional network, $ReLU(\cdot) = max(0, \cdot)$ is a nonlinear activation function.

Our k-core based graph convolution network model enhances the ability of graph representation learning, not only preserves the local proximity of nodes by graph convolutional networks, but also captures the global structure feature by learning the coreness distribution.

B. Graph Decoder

The existence of nodes and edges is modeled as Bernoulli variables in probabilistic graph models, and attributes of nodes are multinomial variables. The decoder is deterministic which is used to predict the adjacency matrix of generated graph. The architecture of the decoder is a multi-layer perceptron, in which the sigmoid activation function is used to compute the adjacency matrix of generated graph. The decoder is given by an inner product between latent variables:

$$p(\mathbf{A}|\mathbf{Z}) = \Pi_{i=1}^{N}\Pi_{j=1}^{N} p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j), \tag{7}$$

with

$$p(A_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^{\top}, \mathbf{z}_j), \tag{8}$$

where $\sigma(\cdot)$ is the logistic sigmoid function.

The generated node attribute matrix is transferred to the input graph, the cross-entropy is as follows:

$$\log p(\mathbf{X}|\mathbf{z}) = \frac{1}{n}\sum_{i} \log X_{i,\cdot}^{T} \tilde{X}_{i,\cdot}, \tag{9}$$

where $\tilde{X}' = \mathbf{C}\tilde{X}$, the $\tilde{X}$ is the predicted adjacency node attribute matrix.

C. Learning Process

We assume that the decoder outputs a graph $\hat{G}$ which is the reconstruct graph , $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ are the adjacency matrix and the node attribute matrix of $\hat{G}$.

The model is trained by minimizing the upper bound on negative log-likelihood [44]. When the regularization is independent of the input space, the reconstruction loss must be designed specifically for each input modality. The reconstruction loss describes the similarity between graph $\hat{G}$ generated by sampling and the input graph G. The cross-entropy is defined as

$$\mathcal{L}_1(\phi, \theta; G) = \mathbb{E}_{q_\phi(\mathbf{z}|G)}[-\log p_\theta(G|\mathbf{z})], \tag{10}$$

where the reconstruction loss can be defined as

$$-\log p(G|\mathbf{z}) = -\lambda_A \log p(\hat{\mathbf{A}}|\mathbf{z}) - \lambda_X \log p(\mathbf{X}|\mathbf{z}). \tag{11}$$

The predicted adjacency matrix $\mathbf{A}' = \mathbf{S}\mathbf{A}$, and the predicted node attribute matrix $\tilde{\mathbf{X}}' = \mathbf{S}\tilde{\mathbf{X}}$, where $\mathbf{S}$ is the absolute value of core matrix difference, i.e. $\mathbf{S} = |\mathbf{C} - \hat{\mathbf{C}}|$. The maximum likelihood estimates are as follows:

$$\log p(\mathbf{X}|\mathbf{z}) = \frac{1}{N}\sum_{1}^{N} \log \mathbf{X}_{i,\cdot}^{\top} \tilde{\mathbf{X}}'_{i,\cdot}, \tag{12}$$

To regularize the code space, we use KL-divergence to sample $\mathbf{z}$ directly from $p(\mathbf{z})$ instead from $q_\phi(\mathbf{z}|G)$ later.

$$\mathcal{L}_2(\phi, \theta; G) = KL(q_\phi(\mathbf{z}|G), p(\mathbf{z})), \tag{13}$$

where $KL(\cdot, \cdot)$ is the Kullback-Leibler divergence function [45] which measures the information loss between two distributions.

We preserve the structural similarity in graph embedding by encoding the neighboring structure into embeddings, i.e. the coreness matrix has been input into embedding. Based on structural equivalence [46], we capture the equivalence structural role between nodes with the same structure identity but no common neighbors in the graph. The node similarity matrix defined by coreness matrix and highly linked node pair usually has a large structure similarity score. According to the definition of k-core and its hierarchical property, we notice that the corenesses of a node's neighbors is highly correlated with that of the node. Therefore, we keep the following structural similarity,

$$\mathcal{L}_3(\phi, \theta; G) = ||\mathbf{C} - \hat{\mathbf{C}}||_F^2 \tag{14}$$

where C and $\hat{C}$ are the coreness matrix of G and $\hat{G}$.

Integrating the three loss equations Eq. 10, Eq. 13, and Eq. 14, the combined loss function is

$$\mathcal{L}(\phi, \theta; G) = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3. \tag{15}$$

In the training process of initial gradient descent, each node aggregates the features (including node features and coreness features) of its neighbors into embedding $\mathbf{z}$, which will preserve the local structural similarity. By iteratively optimizing

the loss function, $\mathbf{Z}$ will recursively aggregate the global structural features. After the above iterations, $\mathbf{Z}$ achieves the fitting. The final embeddings $\mathbf{Z}$ obtained in this way can not only preserve the local features of the graph structure, but also capture the global structural information of the graph.

D. Summary

According to the graph encoder and graph decoder in Section IV-A and Section IV-B, the objective function and learning process in Section IV-C, Combining above together, the learning graph representation $\mathbf{Z}$ is

$$\mathbf{Z} = CGCN(\mathbf{X}, \mathbf{A}), \tag{16}$$

where CGCN is the feature aggregation function mentioned in Section IV-A. The adjacency matrix $\hat{\mathbf{A}}$ of reconstructed graph is

$$\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^{\top}). \tag{17}$$

In the training procedure, we use s to index a training example, and $\hat{s}$ to denote a synthetic example. Then an input graph $G^{(s)}$ is encoded as $Z^{(S)}$, which in turn is decoded to compute the loss $\mathcal{L}$ and reduce the loss in each optimization step.

V. Experiments

In this section, we evaluate our generative model Core-GAE on three tasks: link prediction, graph generation, and node classification. The experimental results and analysis of these three tasks are as follows.

A. Datasets and Setup

The datasets in our experiments include citation networks and bichemical graphs [1][2]. Citation networks represent the relationship between authors and their papers. It is usually used to evaluate the node classification and link prediction tasks of the model. Cora, Citeser and PubMed are citation networks. Biochemical graphs show the relationship between chemical molecules and compound atoms, and chemical bonds are represented as edges. Biochemical graphs are usually used to evaluate the performance of node classification. PPI is a biochemical graph. Table II shows the statistics of the datasets used in the experiments.

TABLE II
Statistics of the Datasets.

Dataset	#Nodes	#Edges	Features
Cora	2708	5429	1433
Citeseer	3327	4732	3703
Pubmed	19717	44338	500
PPI	56944	828716	198

We use the following graph generative models as the experimental comparison of Core-GAE:

- SpectralClustering (SC) [47]: this clustering method cuts the graph, makes the weight sum of edges between different subgraphs as low as possible, and makes the weight sum of edges in subgraphs as high as possible.
- DeepWalk (DW) [48]: it uses random walk to generate node sequence, and use word2vec [49] to learn node representation.
- GAE [36]: It is a unsupervised learning on graph-structured data based on VAE. This model uses a 2 layer GCN as the encoder, and a simple inner product as the decoder.
- VGAE: It uses a simple graph convolution network to encode the graph, and represent the decoded reconstructed graph through the learned graph.
- NF-VGAE [50]: It strengthens the explicit density function and enriches the posterior distribution.
- SIG-VAE [51]: It uses hierarchical variational framework to make neighbors share, and can better use graph structure modeling.
- GCN [52]: it uses a simple graph neural network model to represent and learn graph structure.
- GAT [53]: It is a variant of GCN, which uses attention mechanism to learn the feature of neighbors with different weights.
- MolGAN [23]:It uses an implicit likelihood based node ranking heuristic as graph learning representation.

Parameter Settings. For all the above methods based on graph convolution networks, we set graph convolution network as a two-layer graph convolution network. Adam optimizer [54] is set as the optimization method, the learning rate is set to 0.001 and the weight decay is 5×10^{-4}. For the feature matrix of nodes, we use the one-hot representation. The dimension of the output for embedding for all methods is 128.

B. Link Prediction

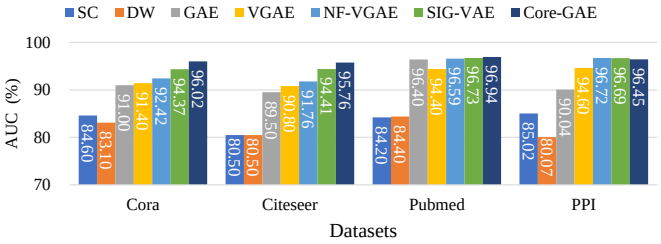
We compare our Core-GAE and other methods to measure the performance of link prediction. The average accuracy (AP) and the curve of area under receiver operating characteristic curve (ROC) (AUC) are the metrics we evaluate in the experiments. We use the 5% edges as the validation set and the 10% edges as the test set. The preprocessing of the datasets is the same as [37], and each method runs 10 times on the test set. For fair comparison, the number of parameters for all methods is similar to the default VGAE.

Fig. 4 shows the results of AUC and AP on the task of link prediction. It can be seen that Core-GAE outperforms known baselines and other comparison methods almost in all datasets. The AUC of our method was more than 95% on each data set, especially on Pubmed, which reached 96.94%, much better than the methods compared in the experiment. Similarly, the AP of our method on each data set is also more than 95%, especially in PPI, which reaches 97.45%. Experimental results show that our algorithm performs well on the task of link prediction compared with other methods.

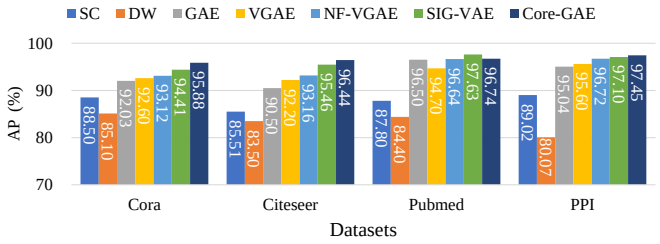
The training curves of AUC and AP for the Core-GAE on four datasets are shown in Fig. 5. In these figures, the x-axis represents the number of training iterations, and the y-axis represents AUC and AP respectively. It can be found that in the first dozens of rounds of training, the error between each

[1]http://konect.cc/networks
[2]http://snap.stanford.edu/index.html

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3085882, IEEE Internet of Things Journal

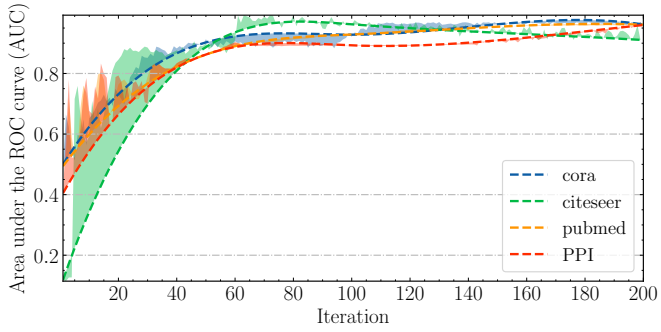


(a) The AUC of link prediction.

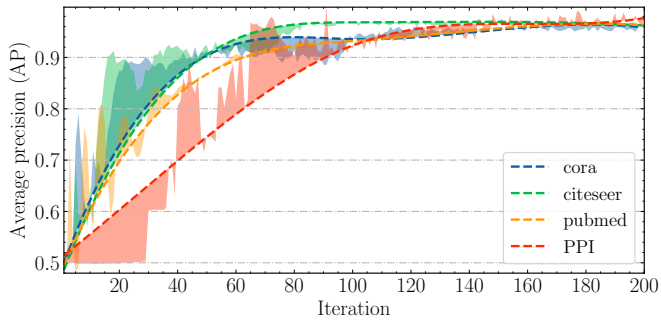


(b) The AP of link prediction.

Fig. 4. The results of link prediction.



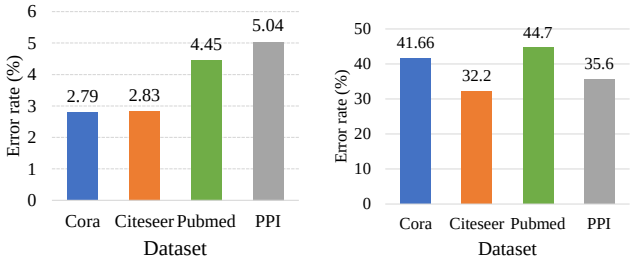
(a) The training curves of AUC.



(b) The training curves of AP.

Fig. 5. The training processing of our algorithm in link prediction.

experiment results of our algorithm is relatively large, and the experimental structure gradually tended to be stable and does not change any more after dozens of rounds. When the number of iterations is enough, the structure of our algorithm can be reflected, and it only needs dozens of rounds to reach stability. For Citeseer, the performance of our algorithm is not as good as the other three datasets in the first 40 rounds of AUC results, but after 60 rounds, our algorithm performs very well on all datasets and became stable. For the data set of PPI, the performance of our algorithm is very unstable in the first 80 rounds in AP tests, but after 100 rounds, the performance of our algorithm on all datasets becomes stable.



(a) Density

(b) Average clustering coefficient

Fig. 6. The percentage of error in the task of graph generation.

C. Graph Generation

To further prove the flexibility and efficiency of Core-GAE, we use the learning representation embedding to construct new graphs. We use two metrics, density and average clustering coefficient, to measure the difference between the original graph and the generated graph. The smaller the difference is between the original graph and the generated graph, the more able our method is to capture the structural properties of graphs. For example, Core-GAE calculates the network parameters of Cora, such as the density $density_o$. Using the autoencoder and learning decoder, a new graph is generated to see if its density $density_n$ is close to the original graph. The percentage error of density is defined as $\frac{|density_n - density_o|}{density_o}$.

The results are illustrated in Fig. 6. Comparing the original graph with the generated graph, the difference of density between the original graph and the generated graph is small. However, in the sparse graph, the average clustering coefficients of the original graph and the generated graph are quite different due to the unobvious structural feature of the sparse graphs. On CiteSeer, our method performed well on both metrics.

D. Node classification

We also applied Core-GAE for node classification task. The results are shown in Fig. 7. By modifying the loss function, including graph reconstruction, our model shows powerful generalization, although it is not specially trained for this task.

The training curves illustrating accuracy of our Core-GAE are shown in Fig. 8. From the result, we can see that our model can achieve good performance within only dozens of iterations on all datasets.

Summary. The experimental results show that our Core-GAE model exhibits good stability and scalability. Compared with the baseline models, our model has good performance in various graph analytic tasks. In addition, our model is suitable

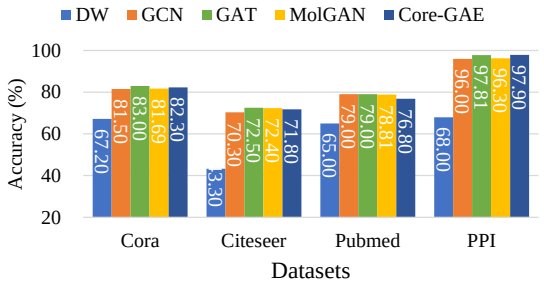


Fig. 7. The AUC of link prediction.

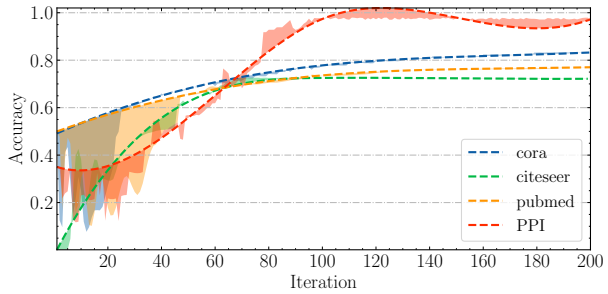


Fig. 8. The training curves of accuracy.

for handling large graph, which is desirable in a real-world implementation.

VI. Conclusion

Considering the rapid development of the Internet of Things and the simulation experiment demands on large-scale generated IoT networks, we propose a coreness based graph autoencoder for IoT network generation. In contrast to previous graph generative models, our proposed Core-GAE can preserve the local proximity similarity and maintain the global structural features. Experimental results illustrate that our approach exhibits much better performance than state-of-the-art methods. Some issues of generating IoT networks still need well studies. In the real world, the networks are evolving and change all the time. Modeling and understanding the generation of dynamic graphs have not been well explored. Furthermore, scalability has been a challenge to the graph generative model. Large networks require large computation and storage to have sufficient information flow. This would limit the scalability of these models. Therefore, we hope that Core-GAE can provide some enlightenment on further research to obtain more advanced generative models.

References

[1] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 2, pp. 766–775, 2020.

[2] A. I. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] N. D. Mueller, J. S. Gerber, M. Johnston, D. K. Ray, N. Ramankutty, and J. A. Foley, "Correction: Corrigendum: Closing yield gaps through nutrient and water management," *Nature*, vol. 490, no. 7419, pp. 254–257, 2012.

[4] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Comput. Networks*, vol. 67, pp. 104–122, 2014.

[5] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[6] Z. Cai and Z. He, "Trading private range counting over big iot data," in *ICDCS*. IEEE, 2019, pp. 144–153.

[7] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial iots," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 968–979, 2020.

[8] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *ICLR*, 2016.

[9] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *14th European Conference of Computer Vision*, vol. 9908, 2016, pp. 776–791.

[10] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," in *ICML*, vol. 70, 2017, pp. 4006–4015.

[11] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*. ACL, 2016, pp. 10–21.

[12] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *ISCA Speech Synthesis Workshop*. ISCA, 2016, p. 125.

[13] N. Jaques, S. Gu, R. E. Turner, and D. Eck, "Tuning recurrent neural networks with reinforcement learning," in *ICLR*. OpenReview.net, 2017.

[14] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *ICLR*, 2016.

[15] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Annual Conference on Neural Information Processing Systems*, 2016, pp. 613–621.

[16] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," in *ICLR*, 2016.

[17] B. D. McKay and A. Piperno, "Practical graph isomorphism, II," *J. Symb. Comput.*, vol. 60, pp. 94–112, 2014.

[18] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning, ICML*, vol. 70. PMLR, 2017, pp. 1263–1272.

[19] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. W. Battaglia, "Learning deep generative models of graphs," in *ICML*, 2018.

[20] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "Graphrnn: A deep generative model for graphs," in *ICML*, 2018.

[21] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *International Conference on Artificial Neural Networks*, vol. 11139. Springer, 2018, pp. 412–422.

[22] T. Ma, J. Chen, and C. Xiao, "Constrained generation of semantically valid graphs via regularizing variational autoencoders," in *Annual Conference on Neural Information Processing Systems*, 2018, pp. 7113–7124.

[23] N. D. Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," in *ICML workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[24] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "Netgan: Generating graphs via random walks," in *ICML*, vol. 80, 2018, pp. 609–618.

[25] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal GCN," in *International Joint Conference on Artificial Intelligence*, 2019, pp. 4419–4425.

[26] V. Batagelj and M. Zaversnik, "An o(m) algorithm for cores decomposition of networks," *CoRR*, vol. cs.DS/0310049, 2003.

[27] P. Erdos and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci.*, vol. 5, no. 1, pp. 17–60, 1960.

[28] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[29] D. J. Watts and S. H. Strogatz, "Collectivedynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[30] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, "An introduction to exponential random graph (p^*) models for social networks," *Soc. Networks*, vol. 29, no. 2, pp. 173–191, 2007.

[31] J. Leskovec, D. Chakrabarti, J. M. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *J. Mach. Learn. Res.*, vol. 11, pp. 985–1042, 2010.

[32] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021.

[33] R. Gómez-Bombarelli, D. Duvenaud, J. M. Hernandez-Lobato, J. Aguilera-Iparraguirre, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS Central Science*, vol. 4, no. 2, 2016.

[34] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, "Grammar variational autoencoder," in *ICML*, vol. 70, 2017, pp. 1945–1954.

[35] C. Yan, S. Wang, J. Yang, T. Xu, and J. Huang, "Re-balancing variational autoencoder loss for molecule sequence generation," in *11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 2020, pp. 54:1–54:7.

[36] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[37] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *CoRR*, vol. abs/1611.07308, 2016.

[38] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Annual Conference on Neural Information Processing Systems*, 2014, pp. 2672–2680.

[39] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks: A survey towards private and secure applications," *ACM Computing Surveys*, 2021.

[40] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *15th International Conference on The Semantic Web*, vol. 10843, 2018, pp. 593–607.

[41] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Annual Conference on Neural Information Processing Systems*, 2017, pp. 5767–5777.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[43] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *CoRR*, vol. abs/1701.07875, 2017.

[44] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations*, 2014.

[45] S. K. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[46] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2357–2366.

[47] S. White and P. Smyth, "A spectral clustering approach to finding communities in graph," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005.

[48] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 701–710.

[49] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *27th Annual Conference on Neural Information Processing Systems*, 2013, pp. 3111–3119.

[50] G. Papamakarios, I. Murray, and T. Pavlakou, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems*, 2017, pp. 2338–2347.

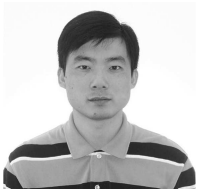
[51] A. Hasanzadeh, E. Hajiramezanali, K. R. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Semi-implicit graph variational auto-encoders," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 711–10 722.

[52] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs 1609.02907, 2016.

[53] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *CoRR*, vol. abs/1710.10903, 2017.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

Qi Luo received his B.S. degree in 2015 from computer science, Northeastern University at Qinhuangdao, China, and M.S. degrees in 2018 from Shandong University, China. Currently, He is pursuing the Ph.D. degree with the School of Computer Science and Technology, Shandong University. His research interests include graph mining and analytics.



Dongxiao Yu received his BS degree in 2006 from the School of Mathematics, Shandong University and the PhD degree in 2014 from the Department of Computer Science, The University of Hong Kong. He became an Associate Professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. Currently he a professor in the School of Computer Science and Technology, Shandong University. His research interests include wireless networks, distributed computing, and graph algorithms.

Yanwei Zheng received the B.S. and M.S. degrees from Shandong Jianzhu University and Shandong University, Jinan, China, in 1999 and 2004, respectively. He joined University of Jinan, Jinan, China, and became a lector from 2004 to 2013. He received his Ph.D. degree from School of Computer Science and Engineering, Beihang University. He joined Shandong University, Qingdao, China in 2019. His research interests include machine learning and computer vision, especially person re-identification.

Hao Sheng received his B.S. and Ph.D. degrees from the School of Computer Science and Engineering of Beihang University in 2003 and 2009, respectively. Now he is a professor and PhD supervisor in the School of Computer Science and Engineering, Beihang University, China. He is working on computer vision,pattern recognition and machine learning.

Xiuzhen Cheng received her M.S. and Ph.D. degrees in computer science from the University of Minnesota Twin Cities in 2000 and 2002, respectively. She is a professor in the School of Computer Science and Technology, Shandong University. Her current research interests include cyber physical systems, wireless and mobile computing, sensor networking, wireless and mobile security. She has served on the editorial boards of several technical journals and the technical program committees of various professional conferences/workshops. She also has chaired several international conferences. She is Fellow of IEEE and a member of ACM.