# Graph Auto-Encoders for Network Completion

Zhang Zhang, Ruyi Tao, Yongzai Tao & Jiang Zhang *

**Abstract.** Completing a graph means inferring the missing nodes and edges from a partially observed network. Different methods have been proposed to solve this problem, but none of them employed the pattern similarity of parts of the graph. In this paper, we propose a model to use the learned pattern of connections from the observed part of the network based on the Graph Auto-Encoder technique and generalize these patterns to complete the whole graph. Our proposed model achieved competitive performance with less information needed. Empirical analysis of synthetic datasets and real-world datasets from different domains show that our model can complete the network with higher accuracy compared with baseline prediction models in most cases. Furthermore, we also studied the character of the model and found it is particularly suitable to complete a network that has more complex local connection patterns.

**1. Introduction.** Networks build underlying structure in many systems and play important roles in science and our daily lives[1, 2]. Thus one way to understand these complex systems is to study the property of the underlying network. However complete network structure usually can't be observed due to measurement errors, privacy concern and other reasons[3, 4, 5]. For example, we can gather online social network easily, but some offline nodes may have huge influence in some times but its hard to gather these offline data. When facing the incomplete network data we need some methods to inference the missing information.

Inferring the complete network structure based on partially observable information has been a fundamental problem in network science and can be beneficial to lots of downstream tasks and applications. Such as node classification[6, 7, 8], graph classification[9, 10], graph generation[11, 12, 13], etc. In most of the network structure inferring works, researchers focus on a task where we can observe all nodes of a network but some edges are missing, this is called link prediction task[14, 15, 16]. One generally used idea to accomplish this task by using structural information such as common neighbors[17, 18] and mutual information[19]. Another way to solve this problem is to use node features to learn which nodes should be connected[20, 21]. Besides Link prediction task, there is another task called network completion, this is harder because the information we can observe is less and the objective is still the same.

In the definition of network completion task, we can only observe some nodes and connections between them, and our objective will be inferring the whole network structure. This problem has been investigated by some previous works. In [22], researchers have proposed a framework called KronEM to solve this problem by using EM algroithms, the basic idea of their work is that self similarity has been a basic property of real world networks, so by observing the part of the network, they can infer a kernel to represent The relationship between

* Authors:
Zhang Zhang, School of Systems Science, Beijing Normal University (3ricccz@gmail.com)
Ruyi Tao, School of Systems Science, Beijing Normal University (taoruyi@mail.bnu.edu.cn)
Yongzai Tao, College of Computer Science and Technology, Zhejiang University (y.tao@zju.edu.cn)
Jiang Zhang*, School of Systems Science, Beijing Normal University (zhangjiang@bnu.edu.cn)

the part and the whole of the network. However, because they use Kronecker product to get the adjacency matrix, the complete network size has to be the power of the kernel size. [23] viewd network completion probelm as network growing problem, from the observed part of the network they learned the growing pattern of the network and that pattern can be generalized to unobserved part. However they needs initial node information such as content of paper in citation network, these information were not easily to gather. And lots of network are not often growing such as economic networks. Similarly, [24] uses the feature of nodes to initialize the estimated network topology, and then complete the network by refining the structure by utilizing node features, labels and distance. DeepNC[25] also focus on network completion problem, they collected a lot of 'similar networks' as training set and use GraphRNN to learn that pattern and generalized to the partial observed network. However the similar network is hard to get in most of the cases. Driven by the dynamics of the network, the nodes of the network will interact to form time series data. [26] mines the information contained in the time series of the observed nodes and completes the network structure of the unobserved part.

In this work we present a new model to solve the network completion problem. The basic idea of our model is that different parts of one network follows the similar connective pattern. We use Graph auto-encoder to learn that pattern and generalize it to unobserved parts. Compared to [23] and [24], our model needs less information. What we only need is part of the adjacency matrix. We don't need similar networks and initial node features neither. Experiments show that our proposed model can complete the network with higher performance.

## 2. Related Work.

2.1. GRAPH AUTO-ENCODER. Graph Auto-Encoder[20, 27] aims to create a graph encoder to embed the graph into a continuous space and we want that embedding vector to represent the graph structure information sufficiently. It has been widely used in many fields because it can represent network structure in low-dimensional space in a trainable way[28, 29, 30]. Usually, one graph auto-encoder consists of two sub-parts, one is encoder, it takes adjacency matrix $A_{n\times n}$ and node features $X \in \mathbb{R}^{N\times k}$ as its input, where $A_{i,j} = 1$ if node $v_i$ connects to node $v_j$ and $A_{i,j} = 0$ otherwise. The encoder maps each node to an embedding vector, thus the output of the encoder will be a matrix $H \in \mathbb{R}^{N\times d}$. Another part of a graph auto-encoder is the decoder. It takes the outputs of the encoder, the vectors of the nodes $H$ as its input and outputs a set of edge probability $\hat{A} \in [0,1]^{N\times\frac{N}{2}}$. The objective of the decoder is to reconstruct the original adjacency matrix as accurate as possible. So for the parameter $\theta$ the loss function can be the cross entropy function as follows:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N}\sum_{j=1}^{\frac{N}{2}} A_{i,j}\log(\hat{A}_{i,j}) + (1-A_{i,j})\log(1-\hat{A}_{i,j}) \tag{1}$$

Usually the encoder is parameterized and the decoder is not. A decoder based Inner Product Decoder and Distance are often considered. Given the embeddings of two nodes $x_i$ and $x_j$, the decoder returns the probability these two nodes connect by caculating the inner product or distance of these nodes embeddings.

Note that if we can reconstruct the adjacency matrix perfectly, it means all the connection information has been stored in the node embeddings $H$. In face that's the objective the graph auto-encoder: generate a set of node embeddings $H \in \mathbb{R}^{N\times d}$(rather than $A \in [0,1]^{N\times N}$) that

express the structure information efficiently.

2.2. GRAPH ISOMORPHISM NETWORK. A lot of Graph Neural Networks(GNN) models can be used as the encoder. In this work we choose Graph Isomorphism Network(GIN)[31]. GIN is a powerful GNN model, it has simple architecture but generates more expressive node embeddings than normal GNN models such as GCN[6] and GAT[35]. Research on the GIN model improves people's understanding of why the GNN model can achieve good results and finds the theoretical upper bound of the GNN model. What's more, GIN reached that upper bound by setting a reasonable aggregate function as follows:

$$h_v^{(k)} = MLP^k((1+\epsilon)h_v^{(k-1)} + \sum_{\mu \in \mathcal{N}(v)} h_u^{(k-1)}) \tag{2}$$

In this function $\epsilon$ is a learnable parameter. It means that we need to sum the embedding vectors of a node and its neighbors and map them through an MLP to get the node representation in the next layer.

2.3. G-GCN MODEL. G-GCN is a network generative model[23], which can be used to solve the link prediction of new nodes of which we can't observe the topologies. The model is trained on a growing network, and the structure is encoded by the node attributes and the known part of the network, so that the edges of the new nodes can be generated.

the input of the model is an undirected network $G(V, E)$, including the observed network $A$ and node $V$ with node attributes $X \in \mathbb{R}^{n \times d}$, and the new nodes $V^{new}$ with attributes $X^{new}$, output the network $A^{new}$ for the whole nodes($V$ and $V^{new}$).

This model is good at dealing with growing networks, such as citation network, but its performance declines when dealing with other networks that do not grow or that grow very little, such as gene regulatory networks.

2.4. KRONEM. KronEM model[22] is proposed to solve the network completion problem for networks that have similar patterns between locally and globally structures. The model optimizes a kronecker kernel from a partial observable network to represent the global-local relationship, the internal parameters of the model are optimized by using the expectation-maximization approach and a scalable, metropolized Gibbs sampling method. However, the KronEM model still has some shortcomings. Firstly, not all networks are self-similar. Moreover, the size of the networks they want to complete must be the power of kernel size, this limits the application of the KronEM model in real-world scenarios.

**3. The Network Completion Problem.** In this section we will introduce the formal definition of network completion problem, and then we will propose our framework based on graph auto-encoder to solve the network completion problem.

3.1. PROBLEM DEFINITION. We consider there is a large undirected network $G(V, E)$ who can not be fully observed since the information about some nodes and their corresponding edges are missing. In contrast, we are able to observe the sub-graph of $G$, which contains some vertices $V_o$ and edges $E_o$ between them. Assume we know the number of missing nodes $N_m$. Our mission is to inference the missing part of the whole network $G$, namely missing nodes $V_m$ and missing edges $E_m$. Fig.1 shows the objective of our task intuitively.

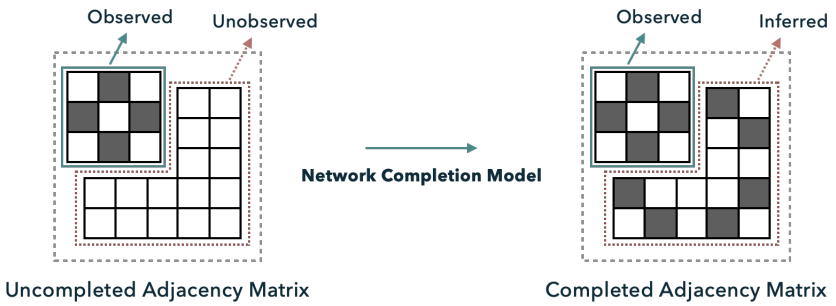


Figure 1. **The network completion problem:** Only some nodes and their connections can be observed on an adjacency matrix. Our aim is to complete the whole network with the information contained in these observable structures.

3.2. PROPOSED FRAMEWORK. The basic idea behind our framework is that we believe that different parts of the network have similar connectivity patterns. For example, there is often a local structure of triangles in a social network of acquaintances (one's friends are usually also friends with each other). In our proposed framework, we will use graph auto-encoder to learn that local connection pattern from partial observed network rand generalize it to unobserved part. Fig.2 shows the working pipeline of our model.

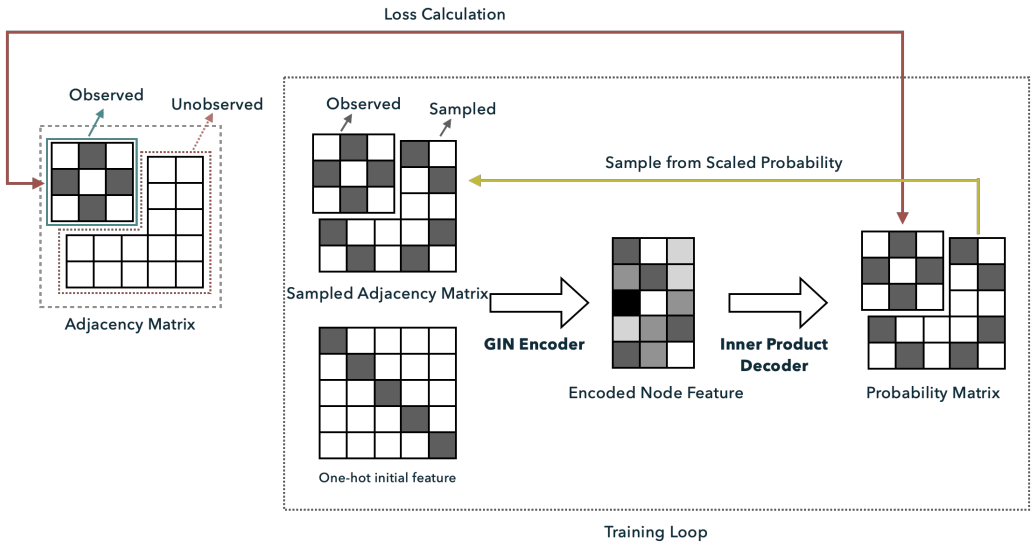


Figure 2. **Overview of our proposed model:** Our model uses GIN to learn the known part of the network structure pattern and iteratively solve the network completion task. Each iteration is divided into the encoding phase and decoding phase, in the encoding phase, the one-hot feature of the node and the completed network from the last iteration are input to the GIN, and the GIN output is embedding of all nodes. During the decoding phase, the decoder decodes the probability matrix according to the embedding of all the nodes. The observable parts of this probability matrix will be used to calculate the loss, and the rest parts will be sampled after scaling to complete the adjacency matrix.

According to the problem definition, we have the information how some observed nodes

connect, namely $A_{N_o \times N_o}$, which is the upper left part of the adjacency matrix. Assume we also know the number of missing nodes $N_m$. At the first start, We will complete the whole adjacency matrix with every unknown part filled with 0 to get a $N \times N$ adjacency matrix $\hat{A}$. Initial feature $X$ of all nodes will be one-hot feature. Then we feed this $\hat{A}$ and nodes' feature $X$ into the GIN encoder to get the encoded node feature $H \in \mathbb{R}^{N \times d}$. We denote the parameter of GIN as $\theta$. This process can be expressd by the Eq.3

$$H = GIN_\theta(\hat{A}, X) \tag{3}$$

We can then feed the encoded node feature into the decoder with Eq.4 and get the decoded probability matrix $P_{N \times N}$. $P_{i,j}$ means the probability that node $v_i$ and node $v_j$ connect.

$$P_{i,j} = \frac{1}{1 + exp(-\langle H^i, H^j \rangle)} \tag{4}$$

Note that probability matrix $P$ is consists of two parts. The upper left-hand $N_o \times N_o$ corner of $P$ means the connection of observed part of the adjacency matrix, this part can be used to calculate the loss and optimize the parameter in GIN encoder. The loss function is defined by cross entropy in Eq.5.

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N_o} \sum_{j=1}^{\frac{N_o}{2}} A_{i,j} \log(\hat{A}_{i,j}) + (1 - A_{i,j}) \log(1 - \hat{A}_{i,j}) \tag{5}$$

Besides the upper left region of $P$. The rest of $P$(inverted L-shaped region) means probability of an unobserved edge exist. We sample from the scaled probability matrix to get the updated adjacency matrix, which means in the next epoch of training, $\hat{A}_{i,j}$ will have the probability $P_{i,j} \times \gamma$ to be 1, in which $\gamma$ is the probability scale factor. It means we will sample the inverted L-shaped region to get the predicted adjacency matrix by the decoded probability matrix and $\gamma$. The reason we put scale factor $\gamma$ here is to control the sparsity of the unobserved part of the predicted adjacency matrix, if this part is too dense, it will has a strong influence on the known part of $A_{N \times N}$ and cause the graph auto-encoder fail to embed the observed nodes efficiently. Thus $\gamma$ can be calculated by Eq.6.

$$\gamma = \frac{N^2 - N_o^2}{N_o^2} \times \frac{\sum_{i<N_o, j<N_o} A_{i,j}}{\sum_{i<N, j<N} P_{i,j} - \sum_{i<N_o, j<N_o} P_{i,j}} \tag{6}$$

In practice, we don't sample the predicted adjacency matrix in each training epoch, because we have to give some time for encoder to learn how to embed the unobserved node such that the sampled L part won't disturb the known upper-left corner of the adjacency matrix. So we set this sample interval as a hyper parameter.

**4. Experiments.** In this section we conduct experiment to exhibit the ability of the proposed model. First we will introduce the baselines we used to compare, including degree product method,AA method and KronEM method. Then we will introduce the metric we use to evaluate the completion performance, namely the AUC and AUPR of unobserved part. We conduct the completion experiment in synthetic network(BA, WS, SBM and Kron) and real-world network in different fileds. Finally we conduct some experiments to study the property of our

proposed model, including how the completion performance change with network size and how hyper parameters influence the ability of the proposed model.

4.1. BASELINES. The baselines were chosen from three different types of network completion algorithm for comparison:

4.1.1. PREFERENTIAL ATTACHMENT. Preferential Attachment(PA) is a common baseline in link prediction task. In this model we calculate the score of one node pair, and then we normalize that score to get the probability these two nodes connect. The score of one node pair can be calculate as eq 7 and eq:8.

$$Score_{x,y} = D(x) \times D(y) \tag{7}$$

$$P_{x,y} = \frac{Score_{x,y}}{\max(Score)} \tag{8}$$

Where $D(x)$ means the degree of node x and $P_{x,y}$ means the probability that node $x$ and node $y$ connect.

In PA model for network completion task, for the reason that one node and its corresponding edges are totally missing, then its impossible to calculate the score of one observed node and unobserved node because the degree of the unobserved node is 0. So we extend the PA model for the completion problem: we only use the degree of the observed node as the score. This is also reasonable because it means that if one observed node has more neighbors, the unobserved will be more likely to link to it.

4.1.2. G-GCN. As we explained in the previous section, G-GCN is a model that solves the network completion problem from the perspective of network growth. The original G-GCN model needed to use node feature information as input, this does not fit our definition of the problem, so here we change the input of this model to the one-hot vector, which is the same as our model. Experiments show that the G-GCN model without node feature is still competitive in some tasks.

4.1.3. KRONEM. The KRONEN model, like our model, only uses observable partial network information instead of node features for network completion. This expands his application scenarios, but it's better suited to completing those networks that follow the Kronecker model, and his performance drops off for networks that don't.

It is important to note that we do not compare this approach on real-world networks, since the number of nodes in the complete graph that can be handled by this model must be a power of kronecker kernel size.

4.1.4. RANDOM-DE. This method uses the same framework as our proposed approach, except that we abandon the GIN model and instead use randomly generated node embeddings of the same size as the decoder's input. The completion result of this method can be interpreted as a random guess, and by comparing it with this method, we can verify whether our model has learned a useful pattern. Besides, we can see that the presence of GIN has what effect on the model, thus acting as an ablation study.

4.2. METRICS. We treated completion of the invert L region of the adjacency matrix as a binary classification task, and we evaluated the performance of our model and baseline models on the basis of two metrics: the area under the ROC curve (AUC) and average precision (AP). We randomly sampled equal numbers of negative and positive edges when evaluating AUC and AP. Moreover, we are particularly interested in evaluating the performance over the part where both endpoints of an edge are in the unobserved part of the network. We denote those values as $AUC_{uu}$ and $AP_{uu}$ respectively.

The network completion algorithm gives us a probability matrix that describes the complete network structure, but we can not use it directly with the adjacency matrix to calculate AUC or AP, this is because the inferred unobserved nodes need to be aligned with the actual unobserved nodes. That's to say, for the probability matrix returned by the network completion algorithm, we need a permutation matrix to reorder the rows and columns corresponding to the unobserved nodes so that the permutated probability matrix resembles the adjacency matrix as much as possible. This process is described in eq9.

$$\arg\min_{p \in P(N!)} ||A - P\hat{A}P^T||_F^2 \tag{9}$$

Where $A$ and $\hat{A}$ means groundtruth complete adjacency matrix respectively, $P$ is the permutation matrix that keeps the $N_o * N_o$ submatrix in the upper left as a diagonal matrix. Therefore, $P\hat{A}P^T$ means reorder the rows and columns between unobserved nodes and remain the observable part unchanged. $|| \cdot ||_F^2$ means the Frobenius norm on matrices. Then the aligning problem can be defined as an optimization problem. Finding the proper p means each of the unobserved nodes we infer has a neighbor structure most similar to that of a real unobserved node.

To solve this problem, we can choose to iterate through all possible unobserved permutations and find the permutation matrix $P$ that works best. However, the number of permutations is $N_m!$, which makes brute force search impossible. Here we introduce the Seed Graph Matching algorithm[], which relaxes the problem by allowing the matrix P to be a doubly stochastic matrix whose entities' values range from 0 to 1. Then they use the conjugated optimization method to find an approximate suitable P in a short time. According to their description, the matching accuracy can be more than 90% when the similarity of two matrices is more than 90% and the number of the nodes to be matched is more than 15. Details can be found in [].

Note that we did not use the Seed Graph Matching method to reorder the probability matrix returned by KronEM, because in order to learn the local and global similarity patterns, the KronEM algorithm will permutate the matrix by itself and returns the aligned probability matrix that it computed.

4.3. COMPLETION PERFORMANCE ON SYNTHETIC GRAPHS. In this section we test the completion performance on synthetic network, concretely we use 4 types of synthetic network generated by some man-made model, namely they are BA network, WS network, FF(Forest fire) network and Kron network. In each network node number is 1024 to fit the need of KronEM algorithm. We randomly choose 25% of nodes and relevant edges to delete. Results are shown in table 1.

In table 1 we can see that in most cases, our model performs better than the comparison model. The Kron network completion model only performs best in the network structure gen-

| Networks | PA | KronEM | Proposed |
|---|---|---|---|
| BA | 0.586 ±0.015 | 0.641 ±0.006 | **0.764 ±0.018** |
| WS | 0.354 ±0.005 | 0.804 ±0.012 | **0.852 ±0.003** |
| Kron | 0.716 ±0.006 | **0.839 ±0.004** | 0.715 ±0.014 |
| FF | 0.793 ±0.009 | 0.621 ±0.004 | **0.801±0.018** |

Table 1. **AUC on unobserved part of synthetic networks:** In this table we show the experiment result of different methods on synthetic networks.

erated by the Kron network generator because it focuses on the global and local similarity of the network. The PA method is only better than the stochastic baseline model in all cases.

4.4. COMPLETION PERFORMANCE ON REAL-WORLD GRAPHS. In this section we test the AUC performance of proposed model on real-world networks. We chose different types of real-world networks to experiment with, their basic information are shown below and the detailed statistics for these networks are shown in table2.

- **Bio_S** is an undirected and weighted networks of gene interactions extracted from C. elegans. The nodes are genes and the edges are Inferred links by genetic interactions. To create an unweighted network, we set all the weights greater than zero to one.
- **Bio_D**, like Bio_S, is also an undirected networks of gene interactions extracted from C. elegans. The difference is the density of the network. From table2, we can see that Bio_S is relatively sparse while Bio_D is relatively dense.
- **Power** is the the western US power network. In this network a node represents a power station, and an edge represents a cable connection between the power stations. This network is undirected.
- **Cora** is a dataset composed of machine learning papers. It is one of the most popular data sets for graph deep learning in recent years, each paper in this dataset has at least one citation. Cora was originally a directed graph. We treat it here as an undirected graph.
- **Co-Author** network represents the co-authorship of researchers in network theory & experiments, where a node is a researcher and if two researchers both made a contribution to one paper, an edge will be created between them.

Note that In this experiment we didn't use KronEM as a compared model for the reason that in KronEM model we have to define a kernel and the number of the nodes has to be the power of kernel size, which is not satisfied in real-world data. For example, if we define the kernel size to be $2 \times 2$, then the adjacency matrix we want to complete has to be 2 to the $k$ power like 4096.

Results are shown in table3. In this experiment, To validate the performance in different regions, i.e edges between observed nodes and unobserved nodes, and edges between unobserved nodes, We generate some data sets with the same number of edges and the same number of edges that do not exist in the corresponding region and measure the performance of the

| Network | Nodes | Edges | Density | Clustering Coefficient |
|---|---|---|---|---|
| Bio_S | 924 | 3239 | 0.0076 | 0.6051 |
| Bio_D | 636 | 3959 | 0.0196 | 0.4712 |
| Power | 1723 | 2394 | 0.0016 | 0.0759 |
| Cora | 2708 | 5278 | 0.0014 | 0.2406 |
| Co-Author | 379 | 914 | 0.0128 | 0.7412 |

Table 2. **Statistics information of different empirical networks.**

| Networks | AUC | Models | | | |
|---|---|---|---|---|---|
| | | PA | G-GCN | Proposed | Random-De |
| Power | All | 48.91 ± 1.4 | **86.18±0.7** | 77.63±0.5 | 82.89±1.0 |
| | Observed-Unobserved | 48.74 ± 0.8 | **92.39±0.6** | 79.37±0.8 | 85.75±0.9 |
| | Unobserved-Unobserved | - | 55.18±0.3 | **62.65±3.3** | 54.16±4.6 |
| Bio_S | All | 72.89 ± 1.3 | 83.23±1.8 | **88.71±2.1** | 70.22±0.7 |
| | Observed-Unobserved | 73.5 ± 1.4 | 88.83±1.7 | **90.16±1.7** | 72.41±0.9 |
| | Unobserved-Unobserved | | 57.19±3.3 | **74.05±3.4** | 54.34±2.2 |
| Bio_D | All | 75.18 ±0.9 | 81.35±1.0 | **85.79±4.0** | 62.43±1.6 |
| | Observed-Unobserved | 77.01 ± 0.8 | 85.86±1.0 | **88.48±2.6** | 63.99±1.8 |
| | Unobserved-Unobserved | - | 57.07±1.1 | **66.71±11.2** | 54.66±2.7 |
| Cora | All | 60.01 ± 0.9 | **86.02±0.9** | 81.30±1.9 | 78.28±1.4 |
| | Observed-Unobserved | 60.13 ± 0.6 | **92.07±0.9** | 82.14±1.9 | 81.28±1.1 |
| | Unobserved-Unobserved | - | 55.58±2.3 | **72.05±5.9** | 53.78±2.1 |
| Co-Author | All | 58.88 ± 2.7 | 85.82±1.7 | **91.61±1.6** | 73.93±1.5 |
| | Observed Uunobserved | 59.07 ± 2.2 | 97.17±1.1 | **93.78±1.8** | 76.27±1.6 |
| | Unobserved-Unobserved | - | 57.36±7.0 | **75.29±7.6** | 60.66±4.1 |

Table 3. **AUC on unobserved part of synthetic networks:** In this table we show the experiment result of different methods on synthetic networks.

model in this region by AUC. In all networks, we randomly removed 20% of the nodes and the corresponding edges, and we ran five repeated experiments to get the mean and the standard deviation to fill the table. To see the results when 10% and 30% nodes are removed or the results measured by AP. please see SI for complete information.

In the table 3 we can see, our proposed model outperforms all of our competitors' model in completing the unknown-unknown part of the adjacency matrix, which means our model is better at modeling connections between unobserved nodes. In the known-unknown part, our model also achieved the best results on two biological networks and co-author networks, but on the power network and the citation network, G-GCN achieved the best performance, this may be due to the fact that G-GCN is inherently suitable for modeling growing networks, whereas power network and citation networks are typical growing networks. In order to further investigate which kind of network is more suitable for our model to complete, we examine the relationship between the model completion performance and the local complexity of the network in the next section.

4.5. PERFORMANCE VS. LOCAL COMPLEXITY. In the encoding stage, we use the GIN model, which gathers neighbor information by message passing. The more times we do message passing, the higher neighbor information can be aggregated by the model. The aggregated information actually represents the local connection structure of a node, that local structure is what our encoder is able to describe. Therefore, compared with the baseline model, our model is able to perform better completion results for networks with more complex local connection structures. Next, we illustrate this point with two experiments.

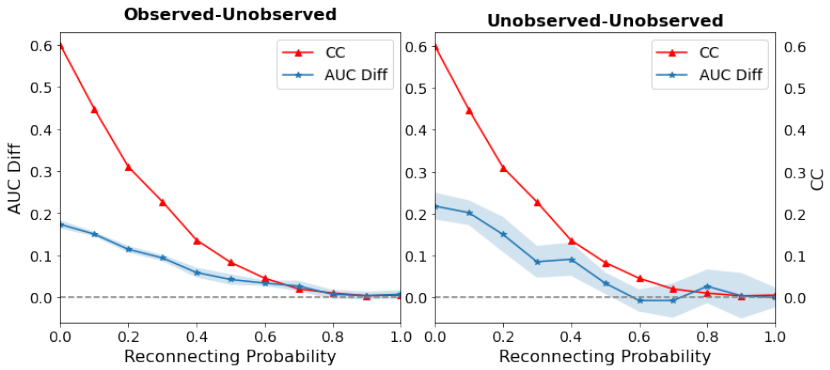


Figure 3. **Performance on W-S network vs. Reconnecting probability**: Indicating the relationship between the AUC(diff) and reconnecting probability on W-S network(blue line). Due to the CC of WS network decreases as the reconnecting probability increases (the red line) , shows that the performance of our model is positively correlated with CC, which is one of the important measurements of local structure complexity.

In the first experiment, we complete the small-world network, and we can continuously observe the effect of local structure complexity on the performance of the network completion by adjusting the reconnecting probability $p$. If $p$ is 0, all nodes are connected to their neighbors, and the neighbors are also connected to each other. In this case, the network has high local complexity. If $p$ is 1, all nodes are connected by the same number of edges randomly, the neighbors of nodes are hardly connected to each other, and the network has low local complexity. Fig3 shows the relationship between the network completion performance and reconnecting probability p. In addition, Fig3 also shows the relationship between the network clustering coefficient(cc) and $p$.

In Fig3, the x-axis is reconnecting probability $p$ of the ws network. This figure has two y-axes, one is cc and another is the AUC difference between the proposed model and the baseline model, in which we replace the output of the GIN encoder with the randomly generated matrix that has the same shape. We can see a clear downward trend. As reconnecting probability $p$ increases, the AUC difference between the proposed model and the baseline model decreases. The AUC difference in the two regions (edges between observed nodes and unobserved nodes and edges between the unobserved nodes) is reduced to about 0 as $p$ increases to 1, which means our proposed model does nothing better than a random guess in this case. this clearly shows that our model is more suitable to complete the network with a more complex local structure. In addition, we can see that CC and also decrease as $p$ increases. This inspired us to use CC to measure the local complexity of the network and to better understand what kind of network our model is suitable for.

In Fig4 , we measure the relationship between the completion effects of different networks and the cc of the network topology. We designed this experiment because the cc of the network can, to some extent, reflect the local complexity of the network. If the neighbors of a node are often connected to each other, then the local structure of the node is complex, whereas if all the neighbors of a node(no matter they are low-order neighbor or high-order neighbor) are not connected to each other, then the local structure of this node is simple. In Fig4, we can see that in most cases of real-world networks, the higher the cc of the network, the better the network completion performance, no matter we complete the connection between the observed nodes and the unobserved nodes. or the connection between the unobserved nodes.

However, we also realize that the cc, which measures the connectivity of a node's one-order neighbors, is not sufficient to measure the local complexity of a network. The experiment of two synthetic network(red star in Fig4) we designed can illustrate this point. They are grid networks and circular networks in which each node is connected to the next first and third node. On these networks, the cc is zero, but all the first-order neighbors of all nodes are connected to their second-order neighbors, so it has a high local complexity, therefore, it has better completion performance.

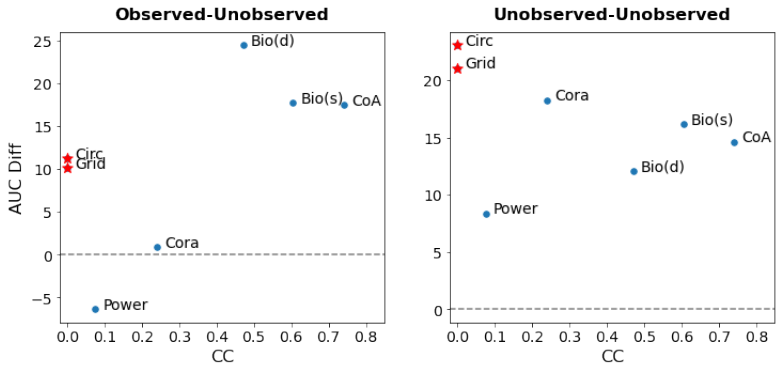


Figure 4. **Performance on Real Networks vs. CC**: X-axis is the Clustering Coefficient of network, y-axis is the performance of our method on the corresponding network. We use the AUC difference between the experiment result and baseline to compare the results among the different networks. The gray dash means this difference equals to zero. Here two outliers(red star) are Grid and Circular with CC = 0 but high complete performance because of their local complex structure.

**5. Conclusion.** In this paper, we propose a model to solve the network completion problem when some nodes and their corresponding edges are missing. This model can learn the local connection pattern of the observable part of the network by Graph Auto-Encode, and generalize it to the unknown region of the adjacency matrix. Experiments show that our model outperforms the competition model in most synthetic networks and real networks, and is especially good at completing the edges between unobserved nodes. By studying the properties of the model, we find that this model is especially good at completing the network with complex local structures.

Of course, this model still has some limitations. In some cases, we can capture the fea-

tures of nodes, for example, nodes on Cora's network are papers, which themselves contain feature information. In some cases, there will be some dynamics between nodes, such as SIR dynamics which describe the temporal dynamics of infectious disease, thus forming the time series data. How to use the pattern of local structure and the features of these nodes and the time series data together to solve the network completion problem will be the direction of our exploration in the future.

**6. Acknowledgements.** We acknowledge the support of the National Natural Science Foundation of China (NSFC) under the grant numbers 61673070. We are grateful for supporting from Save 2050 Project which is sponsored by Swarma Club and X-Order.

## References

[1] P. V. Marsden, "Network data and measurement," *Annual review of sociology*, vol. 16, no. 1, pp. 435–463, 1990.

[2] M. E. Newman, "Communities, modules and large-scale structure in networks," *Nature physics*, vol. 8, no. 1, pp. 25–31, 2012.

[3] G. Cimini, T. Squartini, D. Garlaschelli, and A. Gabrielli, "Systemic risk analysis on reconstructed economic and financial networks," *Scientific reports*, vol. 5, no. 1, pp. 1–12, 2015.

[4] G. Kossinets, "Effects of missing data in social networks," *Social networks*, vol. 28, no. 3, pp. 247–268, 2006.

[5] K. Anand, I. van Lelyveld, Á. Banai, S. Friedrich, R. Garratt, G. Hałaj, J. Fique, I. Hansen, S. M. Jaramillo, H. Lee *et al.*, "The missing links: A global study on uncovering financial network structures from partial data," *Journal of Financial Stability*, vol. 35, pp. 107–119, 2018.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[7] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social network data analytics*. Springer, 2011, pp. 115–148.

[8] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.

[9] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[10] H. Bunke and K. Riesen, "Graph classification based on dissimilarity space embedding," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2008, pp. 996–1007.

[11] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "Graphrnn: Generating realistic graphs with deep auto-regressive models," in *International conference on machine learning*. PMLR, 2018, pp. 5708–5717.

[12] J. Liu, A. Kumar, J. Ba, J. Kiros, and K. Swersky, "Graph normalizing flows," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] Z. Deng, M. Nawhal, L. Meng, and G. Mori, "Continuous graph flow," *arXiv preprint arXiv:1908.02436*, 2019.

[14] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.
[15] P. Wang, B. Xu, Y. Wu, and X. Zhou, "Link prediction in social networks: the state-of-the-art," *Science China Information Sciences*, vol. 58, no. 1, pp. 1–38, 2015.
[16] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 243–252.
[17] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *The European Physical Journal B*, vol. 71, no. 4, pp. 623–630, 2009.
[18] Z. Liu, Q.-M. Zhang, L. Lü, and T. Zhou, "Link prediction in complex networks: A local naïve bayes model," *EPL (Europhysics Letters)*, vol. 96, no. 4, p. 48007, 2011.
[19] F. Tan, Y. Xia, and B. Zhu, "Link prediction in complex networks: a mutual information perspective," *PloS one*, vol. 9, no. 9, p. e107056, 2014.
[20] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.
[21] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in neural information processing systems*, vol. 31, 2018.
[22] M. Kim and J. Leskovec, "The network completion problem: Inferring missing nodes and edges in networks," in *Proceedings of the 2011 SIAM international conference on data mining*. SIAM, 2011, pp. 47–58.
[23] D. Xu, C. Ruan, K. Motwani, E. Korpeoglu, S. Kumar, and K. Achan, "Generative graph convolutional network for growing graphs," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3167–3171.
[24] Q. Wei and G. Hu, "Unifying node labels, features, and distances for deep network completion," *Entropy*, vol. 23, no. 6, p. 771, 2021.
[25] C. Tran, W.-Y. Shin, A. Spitz, and M. Gertz, "Deepnc: Deep generative network completion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
[26] M. Chen, Y. Zhang, Z. Zhang, L. Du, S. Wang, and J. Zhang, "Inferring network structure with unobservable nodes from time series data," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 1, p. 013126, 2022.
[27] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.
[28] Z. Li, J. Li, R. Nie, Z.-H. You, and W. Bao, "A graph auto-encoder model for mirna-disease associations prediction," *Briefings in Bioinformatics*, vol. 22, no. 4, 2021.
[29] T. H. Do, D. M. Nguyen, and N. Deligiannis, "Graph auto-encoder for graph signal denoising," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3322–3326.
[30] Y. Li, C. Meng, C. Shahabi, and Y. Liu, "Structure-informed graph auto-encoder for relational inference and simulation," in *ICML Workshop on Learning and Reasoning with Graph-Structured Data*, vol. 8, 2019, p. 2.
[31] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.
[32] B.-H. Kim and J. C. Ye, "Understanding graph isomorphism network for rs-fmri functional connectivity analysis," *Frontiers in neuroscience*, p. 630, 2020.

[33] Y. Peng, Y. Lin, X.-Y. Jing, H. Zhang, Y. Huang, and G. S. Luo, "Enhanced graph isomorphism network for molecular admet properties prediction," *IEEE Access*, vol. 8, pp. 168 344–168 360, 2020.

[34] Y. Zhang, S. Qiao, L. Sun, Q. W. Shi, W. Huang, L. Li, and Z. Yang, "Photoinduced active terahertz metamaterials with nanostructured vanadium dioxide film deposited by sol-gel method," *Optics Express*, vol. 22, no. 9, pp. 11 070–11 078, May 2014. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-22-9-11070

[35] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017.