entropy

MDPI

Article

Unifying Node Labels, Features, and Distances for Deep Network Completion

Qiang Wei [1,2,*] and Guangmin Hu [1]

[1] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; hgm@uestc.edu.cn
[2] National Key Laboratory of Science and Technology on Blind Signal Processing, Chengdu 610041, China
* Correspondence: weiqiang@std.uestc.edu.cn

Abstract: Collected network data are often incomplete, with both missing nodes and missing edges. Thus, network completion that infers the unobserved part of the network is essential for downstream tasks. Despite the emerging literature related to network recovery, the potential information has not been effectively exploited. In this paper, we propose a novel unified deep graph convolutional network that infers missing edges by leveraging node labels, features, and distances. Specifically, we first construct an estimated network topology for the unobserved part using node labels, then jointly refine the network topology and learn the edge likelihood with node labels, node features and distances. Extensive experiments using several real-world datasets show the superiority of our method compared with the state-of-the-art approaches.

Keywords: network completion; graph convolutional network; node label; node feature; node distance

Citation: Wei, Q.; Hu, G. Unifying Node Labels, Features, and Distances for Deep Network Completion. *Entropy* **2021**, *23*, 771. https://doi.org/10.3390/e23060771

Academic Editor: Miguel A. Fuentes

Received: 9 May 2021
Accepted: 16 June 2021
Published: 18 June 2021

1. Introduction

1.1. Background

Network structures, such as social networks, web graphs, and communication networks, are important to the functioning of complex systems [1,2]. Usually, a complete network structure is a crucial prerequisite for downstream tasks, including node classification and link prediction [3–5]. However, real-world networks tend to be partially observed, with nodes and edges missing due to insufficient resources and privacy protection [3,6,7]. Social networks, such as Twitter and Facebook, have restrictions for crawlers, which makes it impossible for third-party aggregators to collect complete network data. Similarly, when an internet scientist probes the route topology using traceroute, they cannot obtain the structure behind the non-cooperative routes [8]. Thus, the collected network structure is often incomplete, which creates difficulties for downstream analysis.

Our work is motivated by learning the structure of communication networks from passive measurements. In a military context, we may wish to analyze a foreign network inconspicuously [9]. One feasible way is to monitor the packet traffic between the target network and our controlled networks. In an internet reconstruction context, we may wish to obtain a map of networks that have connections with our hosted one for routing strategy optimization [10,11]. In both scenarios, we place passive traffic collectors, and we can collect the user profile (e.g., IP address) [12], hop distance (via TTL) [9,13], and label (via a community) [2] through continuous passive monitoring of the communication, starting from the target network [14]. Passive monitoring provides rich information but there are two limitations: (1) it is often impractical to collect edges or relationships between nodes within the target networks, as their traffic does not pass through our collectors; and (2) there is little control over which targets are measured and, therefore, some data are invariably missing [14].

The difficulty of collecting edges in target networks leads us to the network completion (NC) problem [7]: given a partially observed network structure H from a underlying network G, the goal is to infer the missing unobserved part Z. Network completion can significantly alter our estimates of network-level statistics and node-level structure. In [7], both nodes and edges were missing in Z; however, the nodes are known and only the edges are missing in our scenarios. We consider this setting as a specific network completion problem [15]. Solving the problem is particularly challenging, as all edges in Z are missing. While intuitively similar, NC is fundamentally different from the related well-studied link prediction problem [16]. Edges are missing at random in the link prediction; by contrast, the nodes and edges are missing as a whole in NC, which makes traditional link prediction algorithms unsuitable for NC.

As aforementioned, we focused on solving the specific network completion problem, where node information is known and only edges are missing in Z. The information of nodes, such as labels, features, and distances, is available in our settings [15,17]. It is proven that the side information of nodes is strongly correlated with the underlying network structure [18–22]. Therefore, it is beneficial to integrate side information for NC.

1.2. Related Methods

Model-Based NC: Kim and Leskovec developed KronEM [7], an expectation maximization approach combined with the Kronecker graphs model. They designed a scalable, metropolized Gibbs sampling approach for the estimation of model parameters, as well as inference of the missing part Z. KronEM suffers from three problems [16]: (1) only network topology is considered and side information of nodes is ignored; (2) not all real-world networks follow the Kronecker model; and (3) its speed and performance are not yet satisfactory.

Node-Similarity-Based NC: Other than the missing edges, the node identification and features in G may be available in real settings. Node-similarity-based network completion methods leverage the similarities between node features to infer Z. Matrix completion with decoupled transduction (MC-DT) [15] decouples the completion from transduction to effectively exploit the similarity information. Furthermore, joint node clustering and similarity learning (JCSL) [23] handles the situation, where the node features may be partially missing by computing the node similarities at the cluster level, then jointly co-factorizing the observed adjacency matrix with the cluster-based similarities. However, MC-DT and JCSL have two shortcomings: (1) choosing an appropriate similarity metric is a prerequisite but challenging in practice [24]; and (2) the similarity matrix is exploited linearly, which cannot reflect the nonlinearity between node features and network structures.

Network Structure Learning: Instead of using a similarity graph based on the initial features, another approach in this category is to learn a network structure. Graph neural networks (GNNs) have become the standard toolkit for learning from networks [25,26]. However, most GNNs are designed for a relatively complete network structure, which makes them unsuitable for the network completion problem. Franceschi et al. sampled graph structures from a learnable fully connected structure and employed a bi-level optimization setup for simultaneously learning the GNN parameters and the structure [27]. Chen et al. proposed an iterative method to search for a hidden graph structure that augments the initial graph structure toward an optimal graph for (semi-)supervised prediction tasks [28]. Yu et al. introduced a GCN-based graph revision module for predicting missing edges and revising edge weights via joint optimization [29]. Hao et al. embedded the graph nodes into latent space, and then computed an embedding vector for the unobserved node, with attributes compared to another node's embedding for link prediction [30]. Shin et al. presented Edgeless-GNN for attributed network embedding with edgeless nodes by utilizing a k-nearest neighbor graph based on the similarity of node attributes. However, the existing network structure learning works ignore node labels and distances.

Deep Generative Graph Models: Recent advances in deep generative graph models have furthered progressed in network completion. Inspired by GraphRNN [31], DeepNC

infers the missing parts of a network via deep learning for solving NE-NC [16]. DeepNC first learns a likelihood over the edges via an RNN-based generative graph model by using structurally similar graphs as training data, then inferring the missing parts by applying an imputation strategy for the missing data. Similar to KronEM, DeepNC does not take advantage of the side information of nodes. In addition, DeepNC requires structurally similar graphs for training, which are often difficult to collect in reality; for example, route-level networks are hard to obtain as privacy protection within autonomous systems [32]. Graphite [33] parameterizes variational autoencoders with GNN and uses an iterative graph-refinement strategy inspired by low-rank approximations for decoding. G-GCN [34] produces a unified generative graph convolutional network that learns node embeddings for all nodes by sampling graph generation sequences constructed from H. Graphite and G-GCN are designed for edgeless NC, but neither of them considers node label or distance information, which leaves room for further improvement.

1.3. Present Work

In this work, we address the challenge of network completion using side information. We propose a node-label-, feature-, and distance-based network completion method (LFD-NC), a novel unified deep graph convolutional network that infers the missing edges by leveraging the node label, feature, and distance information. Specifically, we first construct an initial network topology for Z by node labels using a stochastic block model, and then we adopt GNN to obtain a refined estimation of Z using node features; after that, distance constraints are used for pruning the refined network. Lastly, we learn a joint distribution over Z by iteratively performing refinement and pruning.

The contributions of this paper are threefold:

- We formalize NC with side information as a graph refinement problem;
- We propose LFD-NC, a deep graph convolutional-network-based completion method by unifying the observed structure with node label, feature, and distance information;
- We validate LFD-NC through extensive experiments on several real-world networks.

2. Methods

We begin by introducing the network completion problem with side information; then, we propose LFD-NC, a deep graph convolutional-network-based algorithm, to solve the problem.

2.1. Problem Formulation

We assume that there is a true undirected and unweighted network $G(V, E, X, Y)$, where $V = \{v_1, v_2, \ldots v_N\}$ denotes the node set with $|V| = N$, $E \subset V \times V$ denoting the edge set, $X \in \mathbb{R}^{N \times F}$ is the feature matrix for each node in V, and $Y \in \{1, 2, \ldots, C\}^N$ represents the node labels of C classes or communities.

We consider the NC task with node side information over G, where only a part of the topology of G is observed. As illustrated in Figure 1, we have complete knowledge of X, Y, as well as an observed induced subgraph $O(V_O, E_O)$ of G. We denote $A_O = [w_O(u,v)] \in \{0,1\}^{N \times N}$ as the observed adjacency matrix, where $w_O(u,v) = 1$ if $(u,v) \in E_O$; otherwise, $w_O(u,v) = 0$. We also know the shortest-path distance set $D = \{(u, v, d_{uv}) | u \in V_{OD}, v \in V_{ZD}\}$ for some node pairs between $V_{OD} \subseteq V_O$ and $V_{ZD} \subseteq V \backslash V_O$, and we denote $D_u = \{v | (u, v, d_{uv}) \in D\}$ as the observed destination nodes set from source node u. Let $V_Z = V \backslash V_O$ be the node set of the unobserved topology, and let $M_Z = \{V_O \times V_Z\} \cup \{V_Z \times V_Z\}$ represent all possible edges with at least one endpoint in V_Z. The task of NC is to infer the missing edges and non-edges in the unobserved part $Z(V_Z, E_Z)$ where $E_Z \subset M_Z$.

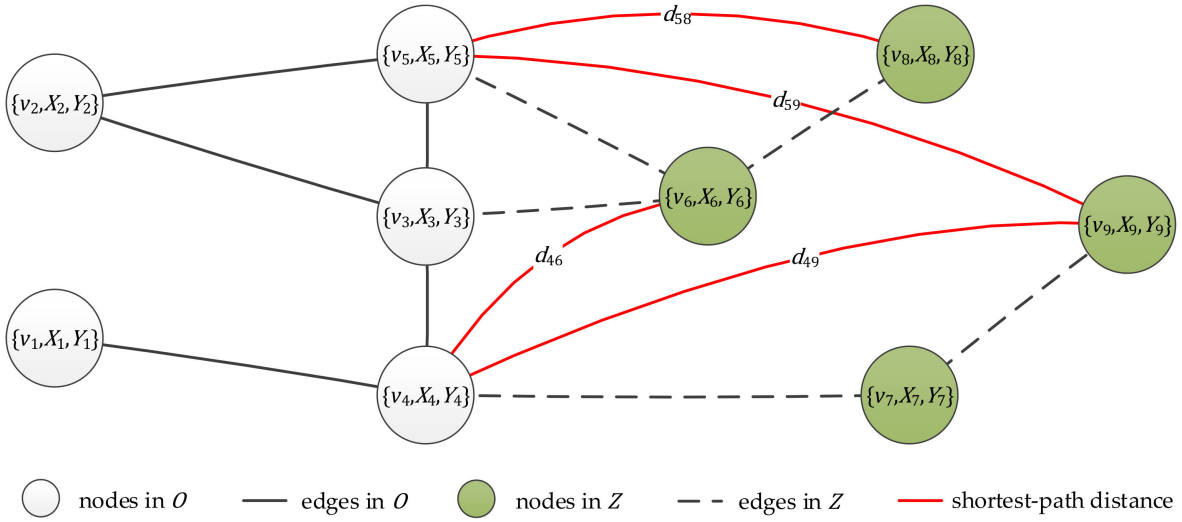

Figure 1. The NC problem with node side information. All nodes $\{v_1, v_2, \ldots, v_9\}$, node feature matrix X, and node label vector Y are known. Edges (black solid lines) in O are observed, but edges (black dashed lines) in the unobserved part Z are missing. Some shortest-path distances (red solid lines) between O and Z are known in addition. To solve NC is to infer the missing edges and non-edges in Z, e.g., edge (v_3, v_6) and non-edge (v_6, v_9).

It is worth mentioning that the observed distance set D considered here is retrieved from passive measurements; therefore, we cannot control the observed shortest-path destination nodes D_u for a given source node $u \in V_{OD}$. In this paper, we further assume that for any source node $u \in V_{OD}$, the destination node set D_u satisfies $D_u \subset V_Z$; that is, we cannot detect all the shortest-path distances from u. Although D indicates deterministic edges and non-edges in Z (see Section 2.5), we treat them as unobserved for the simplicity of the NC problem definition.

Let us model E_Z through the following probability distribution:

$$E_Z \sim P_G(E_Z|O, X, Y, D, \Theta), \tag{1}$$

which is parameterized by Θ. Then, the objective of E-NC is to find the most likely configuration of E_Z.

2.2. Overview of LFD-NC

LFD-NC is motivated by the fact that network topology, node labels, and node features are correlated. Thus, we aimed to find the optimized network topology E_Z that is most consistent with the observed O, X, Y, and D. As there is no closed form for the posterior distribution $P_G(E_Z|O, X, Y, D, \Theta)$ in Equation (1), LFD-NC models it by iteratively refining the network topology. It requires four phases in our computing framework: label-based topology initialization, edge probability learning, distance pruning, and topology refinement. The LFD-NC architecture is shown in Figure 2.

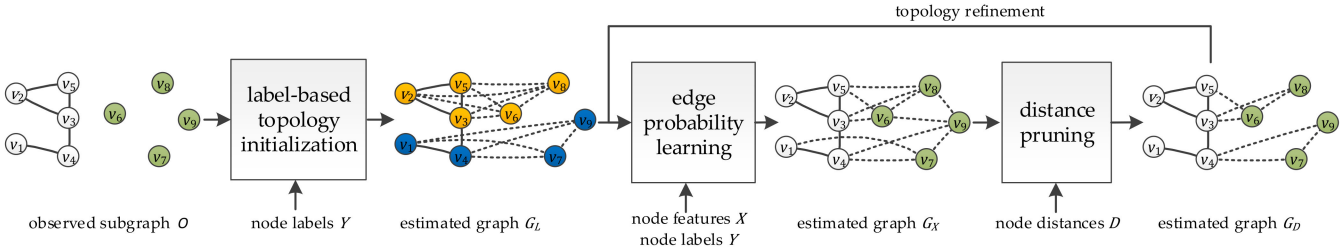

Figure 2. Overview of our LFD-NC. Dashed lines are learned edge probabilities. Nodes with the same color have the same label in G_L.

.

In the label-based topology initialization phase, LFD-NC computes an estimated topology G_L for Z, using only node label information Y. Then, in the edge probability learning phase, we take G_L, node feature X and node label Y as the inputs to learn a new edge probability $P_X(u,v)$ for $(u,v) \in M_Z$, which leads to a better topology G_X. Notably, learning P_X is a typical link-prediction task, where standard methods such as GNN [35,36] or improved methods with label propagation [37–39] can be directly applied as aforementioned.

We then prune edges that should not exist on the basis of node distance constraints D. Lastly, we refine the estimated topology G_X by iteratively performing edge probability learning and distance pruning.

From a node-embedding perspective, we first map each node in G to a low-dimensional vector in LFD-NC's first two phases, then we learn a pairwise decoder to predict the edges and non-edges in Z in the edge probability learning phase. Lastly, we refine the network topology in the distance pruning phase, and recalculate the node embeddings correspondingly using topology refinement.

Next, we elaborate the details of each step.

2.3. Label-Based Topology Initialization

NC suffers from the cold start problem [15,34], i.e., there are no prior connections for the nodes in Z. Therefore, standard GNN methods, such as GCN [35] and GAT [40], cannot be directly applied for NC since the missing edges block message passing and aggregation between O and Z.

We present node-label-based topology initialization to overcome the cold start problem in NC, which is motivated by two insights. Firstly, the community structure is positively correlated with the underlying network topology [21]. Most complex networks show a community structure, i.e., blocks of nodes that have a high density of edges within them, and a lower density of edges between them. Community structure is often detected on the basis of the known underlying network topology. Here, we take the opposite direction and treat the community structure as a good initial estimation of the unobserved network topology. Secondly, the label information can be integrated to improve the performance of node embedding. It is proven that unifying label propagation and GNN overcomes the over- or under-smoothing issue of GNN [37,38,41]. In this paper, we treat the known labels Y as the optimized result of the label propagation procedure, and then find the corresponding topology.

We initialize edges in M_Z using the stochastic block model (SBM) [3,42,43], which is widely used to model communities in complex networks by modulating the intra- and extra-block connections. Specifically, the edge probability in M_Z is determined by the following:

$$P_L(u,v) = p_{Y(u),Y(v)}, \tag{2}$$

where $\left[p_{Y(u),Y(v)}\right] \in [0,1]^{C\times C}$ is the probability of an edge between communities. The estimated edge weight in M_Z is calculated as follows:

$$w_L(u,v) = \alpha_L P_L(u,v), \tag{3}$$

where α_L is a parameter that controls the strength of P_L in the estimated graph G_L. If we set $w_L(u,v) = 0$ for $u,v \in V_O$, and denote $W_L = [w_L(u,v)] \in \mathbb{R}^{N\times N}$ as the SBM-estimated matrix, then the adjacency matrix of G_L can be represented by the following:

$$A_{G_L} = A_O + W_L. \tag{4}$$

We show that A_{G_L} can be the underlying topology for label propagation. The label propagation procedure in iteration k can be formulated as follows [37]:

$$Y^{(k+1)} = \widetilde{D}^{-1} A_G Y^{(k)}, \tag{5}$$

where $A_G = \left[a_{ij}\right] \in [0,1]^{N\times N}$ is the partial unknown adjacency matrix of G, and $\widetilde{D}$ is a diagonal matrix with $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$. We have $Y^{(k+1)} = Y^{(k)} = Y$ in our settings; hence Equation (5) holds when $A_G = A_{G_L}$, which indicates that A_{G_L} is a valid solution of Equation (5). Therefore, A_{G_L} is consistent with the label propagation.

2.4. Edge Probability Learning

Edge probability learning aims to obtain a better network topology from G_L, X and Y, and we treat it as a link prediction task. After the label-based topology initialization phase, we directly exploit network embedding techniques to find the proper function f as follows:

$$H = f\left(A_{G_L}, X, Y\right), \tag{6}$$

where H is the node embedding matrix.

Many existing methods can be used to obtain H [35–39]. In this paper, we adopt GCN [35] to model f. In GCN, the hidden representations for each layer can be obtained by the following:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-\frac{1}{2}} \widetilde{A}\, \widetilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right), \tag{7}$$

where $\widetilde{A} = A_{G_L} + I_N$ is the adjacency matrix of G_L with added self-loops, I_N is the identity matrix, $\widetilde{D}$ is a diagonal matrix with $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$, $W^{(l)} \in \mathbb{R}^{F^{(l)}\times F^{(l+1)}}$ is a layer-wise learnable weight matrix, $\sigma(\cdot)$ denotes an activation function such as ReLU [44], and $H^{(l)}$ is the node embedding of layer l with $H^{(0)} = [X\Big|Y]$, where $[X|Y]$ is a concatenation of X and one-hot label indicators.

We consider a two-layer GCN as our forward model, and the final embedding is the following:

$$H = \hat{A}\; ReLU(\hat{A}[X\Big|Y]W^{(0)})W^{(1)}, \tag{8}$$

where $\hat{A} = \widetilde{D}^{-\frac{1}{2}} \widetilde{A}\, \widetilde{D}^{-\frac{1}{2}}$. The weight matrices $W^{(0)}$ and $W^{(1)}$ are calculated by minimizing the cross-entropy errors of labeled edges in O:

$$\mathcal{L}_{cross-entropy} = -\sum_{(u,v)\in E_O} \log(P_X(u,v)) - \sum_{(u,v)\notin E_O} \log(1 - P_X(u,v)). \tag{9}$$

Due to the sparse nature of real-world networks, there are only a small number of edges in all node pairs; thus, we generate $|E_O|$ non-edges via random sampling.

The edge probability in M_Z then takes the following simple form:

$$P_X(u,v) = \text{sigmoid}\left(H_u H_v^T\right). \tag{10}$$

As a function Equation (10), we provide the realization of Equation (1) as follows:

$$P_G(E_Z\Big|O,X,Y,D,\Theta) = \prod_{(u,v)\in E_Z} P_X(u,v) \prod_{(u,v)\notin E_Z} (1 - P_X(u,v)). \tag{11}$$

We denote $P_X = [p_X(u,v)] \in \mathbb{R}^{N\times N}$ as the link likelihood matrix, and set $p_X(u,v) = 0$ for $u, v \in V_O$; then, the adjacency matrix of G_X can be represented by the following:

$$A_{G_X} = A_O + P_X. \tag{12}$$

2.5. Distance Pruning and Topology Refinement

Distance pruning and topology refinement aim to further improve the performance of node embedding. The distance constraint D indicates the existence of edges and non-edges between some node pairs, whereby clamping the edge probability of these node pairs leads to a clearer network topology G_D. Then, we take G_D instead of G_L, and repeat the edge probability learning process to gradually refine the node embedding matrix H.

Given the distance constraint D, we may calculate two deterministic sets: an edge set $E_D \subset M_Z$ and a non-edge set $E_D^{\varnothing} \subset M_Z$. The calculation is based on Observation 1 and Observation 2 [11].

Observation 1. *For any given nodes u, v and w in an undirected and unweighted graph $G(V, E)$, if $|d_{uv} - d_{uw}| \geq 2$, then $(w, v) \notin E$ holds.*

Observation 2. *Let $L_u^i = \{v | d_{uv} = i, v \in N\}$ be the sets of nodes with the same distance i from u. For two given nodes $v \in L_u^i$ and $w \in L_u^{i+1}$, if for any node $x \in L_u^i \backslash \{v\}$, $(x, w) \notin E$, then $(v, w) \in E$ holds.*

Note that Observation 2 needs u observed distances to all the other nodes in G, which cannot be met under our assumption. Therefore, E_D only contains the observed direct neighbors of the distance monitor nodes V_{OD}.

After the calculation of E_D and $E_D^{\varnothing}$, we clamp the probability of edges in E_D to 1, and the probability of non-edges in $E_D^{\varnothing}$ to 0. Let $M_{non-edge} = [m(w, v)] \in \{0, 1\}^{N \times N}$ denote the non-edge mask matrix where $m(w, v) = 1$ if $(w, v) \in E_D^{\varnothing}$. Then, the distance pruning process can be represented as follows:

$$A_{G_X} \left[M_{non-edge} \right] = 0. \tag{13}$$

Then, we assign the adjacency matrix A_{G_D} of G_D as the masked A_{G_X}. We ignore E_D in LFD-NC as $|E_D| \ll |E_D^{\varnothing}|$.

We summarize our algorithm in Algorithm 1.

The time complexity of LFD-NC is the same as that of GCN. The complexity of line 1 in Algorithm 1 is $\mathcal{O}(|M_Z|)$. In GCN, it is usually satisfied that $F > F^{(1)} \geq F^{(2)}$; thus, the complexity of line 3 is $\mathcal{O}(N^2 F + N F^2)$. The complexity of lines 4 and 5 is $\mathcal{O}(|M_Z|)$. Since $|M_Z| < N^2$, the total complexity of LFD-NC is dominated by GCN.

Algorithm 1. LFD-NC

Input: node features X, non-edge mask $M_{non-edge}$, observed graph matrix A_O, SBM estimated matrix W_L, and topology refinement round R.
Output: estimated graph A_{G_D}.

1.	$A_{G_L} = A_O + W_L$	//label-based topology initialization by Equation (4)
2.	*forrin* $[1, 2, \ldots, R]$	//topology refinement
3.	$H = f(A_{G_L}, X)$	//node embedding by Equation (6)
4.	$A_{G_X} = A_O + P_X$	//link prediction by Equation (12)
5.	$A_{G_X} \left[M_{non-edge} \right] = 0$	//distance pruning by Equation (13)
6.	$A_{G_L} = A_{G_D} = A_{G_X}$	//update A_{G_L}
7.	*end for*	
8.	*output* A_{G_D}	

3. Experiments

We conducted a series of experiments. First, we evaluated the performance of LFD-NC and compared it with the state-of-the-art network completion methods. Then, we analyzed the impacts of the four phases.

3.1. Experimental Settings

3.1.1. Datasets

We evaluated the performance of LFD-NC on eight real-world network datasets. The details of the eight datasets are presented below.

Actor is a film actor network [45]. Each node corresponds to an actor, and the edge between two nodes denotes co-occurrence on the same Wikipedia page. Node features correspond to some keywords in the Wikipedia pages. Nodes are classified into five categories in terms of words in the actor's Wikipedia entry.

Cornell, Texas, and Wisconsin are three small local web networks from WebKB [45], which is a webpage dataset collected by Carnegie Mellon University from computer science departments at various universities. Nodes and edges represent web pages and hyperlinks, respectively. The feature of each node is the bag-of-words representation of the corresponding page, and the label of each node is manually classified into five categories: student, project, course, staff, and faculty.

Cora, Citeseer, and PubMed are three classic citation networks [46]. Nodes represent scientific papers, and edges represent citation relationships. Node features correspond to the bag-of-words representation of the paper and the label of each node represents one of the academic topics.

WikiCS is a Wikipedia web network [47]. It consists of nodes corresponding to computer science articles, with edges based on hyperlinks. Node features are derived from the text of the corresponding articles. Nodes are labeled into 10 classes representing different branches of the field.

The dataset statistics are shown in Table 1. We only focused on the largest connected component for each network in this paper.

Table 1. Dataset statistics.

Dataset	Nodes	Edges	Classes	Features
Actor	7600	33,544	5	931
Cornell	183	295	5	1703
Texas	183	309	5	1703
Wisconsin	251	499	5	1703
Cora	2708	5429	7	1433
Citeseer	3327	4732	6	3703
PubMed	19,717	44,338	3	500
WikiCS	11,701	216,123	10	300

3.1.2. Baselines and Evaluation Metrics

The baselines were chosen from five different types of network completion algorithm for comparison:

- SBM [42] only uses node labels, whereby the symmetric $C \times C$ matrix of edge probability $\left[p_{Y(u),Y(v)}\right]$ is estimated from the observed O;
- KronEM [7] only uses the network graph structure and ignores node features, node labels, and node distances;
- MC-DT [15] employs both the pairwise similarity of node features and the network graph structure, as well as ignores node labels and node distances. The similarity information is utilized by matrix factorization in a linear way;
- MLP-NC [48] considers node features and the network graph structure, as well as ignores node labels and node distances. Unlike MC-DT, MLP-NC directly learns a non-linear similarity metric;
- G-GCN [34] also considers node features and the network graph structure, as well as ignores node labels and node distances. Unlike MLP-NC, G-GCN adopts a generative graph convolution model.

We treated the prediction of missing edges in Z as a binary classification, and we evaluated the performance of LFD-NC 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.

3.1.3. Implementation Details

We generated $O(V_O, E_O)$ by breadth-first search (BFS) traversal from a randomly selected node and split $E \backslash E_O$ equally for validation and testing. Before training, node features X were normalized, and then extended by concatenating the one-hot encoding of Y for a fair comparison with MC-DT, MLP-NC and G-GCN.

We adopted the realization of KronEM in SNAP [49] and kept all parameters as the default, except for the initial Kronecker matrix, which we set to $[0.9, 0.7; 0.7, 0.2]$ for symmetry. We implemented MC-DT, using the SciPy Python library, while the feature similarity matrix was computed by the cosine metric, and we set the number of eigenvectors s to 20. We executed G-GCN from the officially released code, and kept all parameters as the default, except for $F^{(1)}$ and $F^{(2)}$. We implemented MLP-NC and LFD-NC in PyTorch and open-sourced them at https://github.com/weiqianglg/LFD-NC.

For LFD-NC, we constructed D by randomly selecting a fixed number of source nodes for V_{OD}; then, we randomly split V_Z into $|V_{OD}|$ disjoint subsets, where each subset corresponded to one source node. The topology refinement round was set to $R = 4$. We adopted the planted partition model in SBM initialization, and we fixed $p_{Y(u),Y(v)} = 1$ if nodes u, v had the same label; otherwise, $p_{Y(u),Y(v)} = 0$ in Equation (2). For G-GCN, MLP-NC, and LFD-NC methods, we set the hidden dimension as $F^{(1)} = 64$ and the final dimension as $F^{(2)} = 32$, and we used the Adam optimizer with a learning rate of 0.01 to train the three deep-learning models for all the datasets. We also applied an early stop strategy over AUC for the validation set, with the patience set to 20 epochs.

3.2. Completion Performance

3.2.1. Comparison with State-of-the-Art Methods

Tables 2 and 3 summarize the comparison results for the eight real-world network datasets. We kept $|V_{OD}| = 16$ and $|E_O|/|E| = 0.8$, and we set $\alpha_L = 10^{-4}$ for Cornell, Cora, Pubmed, and WikiCS, $\alpha_L = 6 \times 10^{-2}$ for Actor and Texas, $\alpha_L = 8 \times 10^{-2}$ for Wisconsin, and $\alpha_L = 1 \times 10^{-1}$ for Citeseer. The mean and the confidence interval of AUC and AP were measured by 10 random BFS samplings. LFD-NC outperformed all other comparative models in the eight datasets. These results demonstrate that incorporating node label and distance constraints into GNN models significantly improves the solution of an NC problem.

Table 2. Comparison of test set AUC score (%) with state-of-the-art methods. The best results are marked in bold.

Method	Actor	Cornell	Texas	Wisconsin	Cora	Citeseer	PubMed	WikiCS
SBM	49.7 ± 0.6	45.1 ± 5.8	65.2 ± 8.0	52.7 ± 7.0	88.9 ± 1.5	80.7 ± 2.0	76.1 ± 1.6	83.1 ± 0.8
KronEM	54.0 ± 1.2	52.0 ± 12.3	59.3 ± 9.7	51.5 ± 9.2	50.9 ± 1.4	49.3 ± 2.1	57.6 ± 1.7	62.2 ± 2.9
MC-DT	50.6 ± 0.7	48.9 ± 9.4	58.8 ± 10.5	72.7 ± 2.9	91.6 ± 1.6	87.7 ± 1.6	89.4 ± 0.8	92.3 ± 0.9
MLP-NC	51.6 ± 1.1	43.1 ± 9.1	41.8 ± 9.4	66.7 ± 8.0	90.1 ± 0.9	85.6 ± 1.6	88.6 ± 1.2	92.3 ± 1.2
G-GCN	50.4 ± 0.7	53.8 ± 4.5	38.9 ± 7.8	60.0 ± 6.0	93.2 ± 0.2	88.7 ± 0.2	88.7 ± 0.2	90.7 ± 0.5
LFD-NC	**72.1** ± 1.0	**85.7** ± 2.3	**88.4** ± 2.8	**89.4** ± 4.2	**97.1** ± 0.6	**96.0** ± 0.7	**93.7** ± 0.9	**92.5** ± 0.9

KronEM uses only the observed network topology for completion, and SBM uses only node labels, which led to them performing the worst. MC-DT, MLP-NC, and G-GCN performed similarly to each other in general, but they consider neither node labels nor distance constraints; therefore, LFD-NC outperformed almost all of them. For example, LFD-NC achieved about 21.9–23.2% AUC and 16.1–22.5% AP absolute improvements, compared with the second-best methods in Cornell and Texas. These improvements were contributed by both node label information and distance constraints; in Cora and Citeseer,

LFD-NC achieved 5.5–7.3% AUC and 5.2–8.4% AP absolute improvements, compared with the second-best methods. These improvements were mainly contributed by distance constraints. We systematically studied the impact of node labels and distance constraints, as presented below.

Table 3. Comparison of test set AP score (%) with state-of-the-art methods. The best results are marked in bold.

Method	Actor	Cornell	Texas	Wisconsin	Cora	Citeseer	PubMed	WikiCS
SBM	49.9 ± 0.5	49.4 ± 3.9	68.7 ± 8.3	56.7 ± 4.5	85.8 ± 2.1	79.2 ± 1.8	71.3 ± 1.7	81.6 ± 0.9
KronEM	53.0 ± 1.3	56.0 ± 11.4	59.9 ± 7.6	55.0 ± 7.9	51.5 ± 1.4	50.0 ± 1.6	55.3 ± 1.6	59.9 ± 2.5
MC-DT	51.1 ± 0.7	54.6 ± 7.3	57.3 ± 7.3	73.4 ± 4.0	89.0 ± 2.3	86.3 ± 1.9	88.3 ± 0.8	91.5 ± 1.1
MLP-NC	52.2 ± 1.2	52.6 ± 8.9	47.5 ± 6.2	67.1 ± 8.1	87.3 ± 1.3	81.5 ± 2.3	86.5 ± 1.4	92.0 ± 1.3
G-GCN	51.4 ± 0.9	60.8 ± 5.4	46.2 ± 4.8	59.6 ± 5.8	91.4 ± 0.2	86.2 ± 0.2	86.2 ± 0.2	90.6 ± 0.5
LFD-NC	**66.7** ± 1.4	**83.3** ± 3.5	**84.8** ± 5.4	**87.5** ± 6.4	**96.6** ± 0.7	**94.6** ± 1.2	**92.3** ± 1.2	**92.3** ± 1.0

3.2.2. Impact Analysis of Node Labels

Node labels are integrated into the initial topology by the SBM-estimated matrix, which is controlled by the parameter α_L, as shown in Equations (3) and (4). The SBM estimation enhances the connections for nodes in the same class, but it creates noisy edges that do not actually exist. Therefore, we need to properly set the parameter α_L.

Figure 3 shows the impact of parameter α_L on AUC and AP in Actor and Cora. The importance of α_L clearly varied with the dataset. A properly chosen α_L produced 5% absolute AUC and AP improvements, compared with $\alpha_L = 0$ in Actor, whereas there was no improvement in Cora.

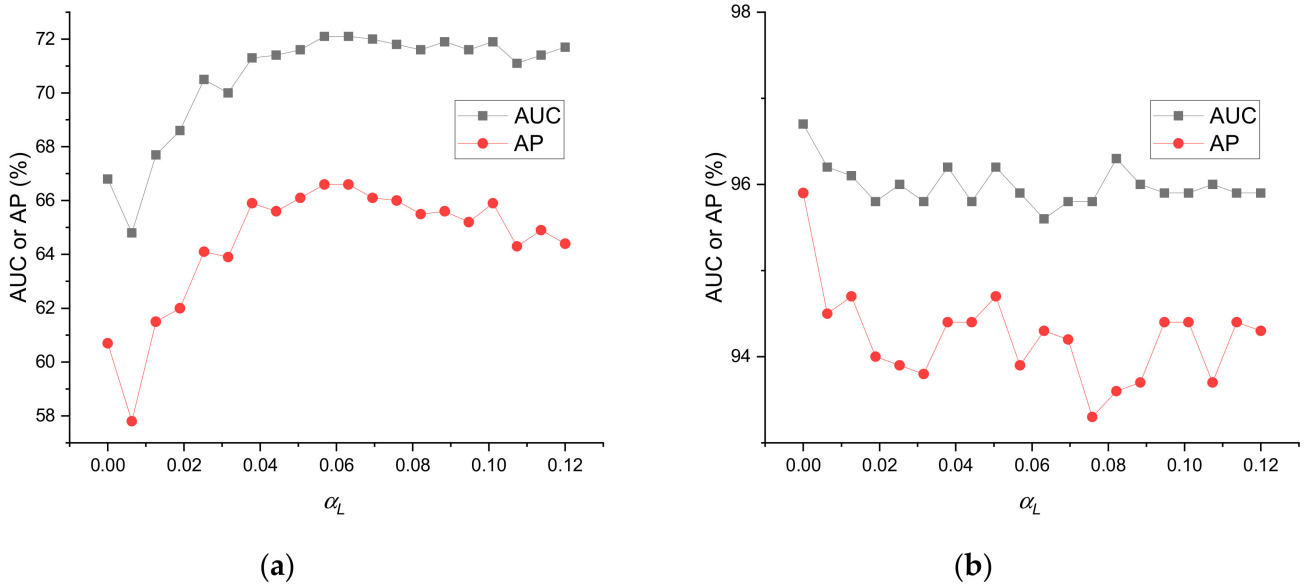

Figure 3. Impacts of the parameter α_L on AUC and AP with $|E_O|/|E| = 0.8$ for (**a**) Actor and (**b**) Cora.

The effectiveness of the label-based topology initialization was affected by the correlation between the node features and edges. When α_L was close to zero, LFD-NC degenerated into a MLP model, which did not use label information; when α_L became large, more noisy edges were added into the initial topology A_L, which eventually degraded the performance. Therefore, a small α_L produced good results in Cora, where the edge likelihood was mainly determined by the node features; however, in Actor, neither small nor large α_L produced the best results.

3.2.3. Impact Analysis of Distance Constraints

Figure 4 presents the impacts of the distance constraints on AUC and AP. A higher deterministic edge rate $\left|E_D^{\varnothing}\right|/|M_Z|$ achieved about 5.3–14.4% AUC and 5.5–12.6% AP absolute improvements, compared with zero monitors ($\left|E_D^{\varnothing}\right|/|M_Z| = 0$). Distance pruning restricted the probability of edges in Z and reduced the uncertainty of the estimated topology; therefore, the AUC and AP of LFD-NC increased with $\left|E_D^{\varnothing}\right|/|M_Z|$.

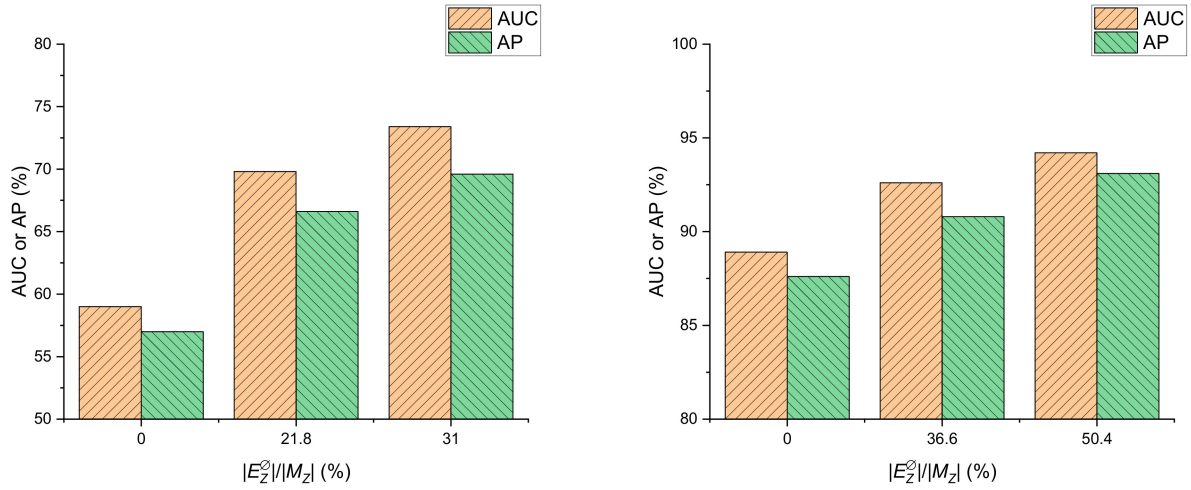

Figure 4. Impacts of the distance monitor number on AUC and AP with $|E_O|/|E| = 0.5$ for (**a**) Actor and (**b**) Cora.

3.2.4. Ablation Study

LFD-NC solves the NC problem in four sequential phases; here, we performed an ablation study in LFD-NC on Actor and Texas datasets, as shown in Table 4. Compared with the standard LFD-NC, removing the label-based topology initialization phase resulted in the highest decrease in AUC and AP (6%); removing the topology refinement phase resulted in the highest decrease in AUC and AP (11%); and removing the distance pruning phase resulted in the highest decrease in AUC and AP (>26%). The experiments show that the four phases are all necessary for LFD-NC, and the distance pruning phase is particularly important.

Table 4. An ablation study of LFD-NC (%). LTI indicates label-based topology initialization, EPL indicates edge probability learning, DP indicates distance pruning, and TF indicates topology refinement.

LTI	EPL	DP	TF	AUC on Actor	AP on Actor	AUC on Texas	AP on Texas
✓	✓	✓	✓	72.1 ± 1.0	66.7 ± 1.4	88.4 ± 2.8	84.8 ± 5.4
	✓	✓	✓	66.4 ± 1.7	60.1 ± 1.6	86.7 ± 6.7	81.3 ± 8.9
✓	✓	✓		70.7 ± 1.4	65.4 ± 2.1	79.1 ± 8.0	74.1 ± 10.8
✓	✓		✓	60.4 ± 0.8	57.3 ± 0.9	50.9 ± 3.3	58.3 ± 6.5

4. Conclusions and Future Work

We presented LFD-NC, a unified deep graph convolutional network for network completion. LFD-NC integrates node label, feature, and distance information through a graph refinement framework. Experiments on eight datasets demonstrated that our model outperforms state-of-the-art baseline methods.

Our work can be easily extended to directed graphs and multigraphs. To treat directed graphs, the only necessary change is to perform directed node embedding in the edge probability learning phase. To treat multigraphs, we take each type of relationship individually, and then combine all the inferred results to achieve a final completion.

We note three possible directions for future work. Firstly, our proposed model assumes that node labels Y and node features X are completely known. An interesting direction would be to perform completion when parts of X and Y are missing and noisy. Secondly, LFD-NC has a long training time for large-scale networks; thus, reducing the time complexity is also a possible future research direction. Thirdly, we integrated distance constraints from randomly selected monitors; therefore, an effective monitor placement strategy can be designed for further performance improvement.

Author Contributions: Conceptualization, Q.W.; methodology, Q.W.; software, Q.W.; writing—original draft preparation, Q.W.; writing—review and editing, Q.W. and G.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Barabási, A.-L. Networks at the heart of complex systems. In *Network Science*; Cambridge University Press: Cambridge, UK, 2016.
2. Newman, M.E.J. Communities, modules and large-scale structure in networks. *Nat. Phys.* **2012**, *8*, 25–31. [CrossRef]
3. Hanneke, S.; Xing, E.P. Network completion and survey sampling. In Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, Clearwater Beach, FL, USA, 16–19 April 2009; PMLR: Clearwater Beach, FL, USA, 2009; Volume 5, pp. 209–215.
4. Hric, D.; Peixoto, T.P.; Fortunato, S. Network structure, metadata, and the prediction of missing nodes and annotations. *Phys. Rev. X* **2016**, *6*. [CrossRef]
5. Newman, M.E.J. Network structure from rich but noisy data. *Nat. Phys.* **2018**, *14*, 542–545. [CrossRef]
6. Huisman, M.; Krause, R.W. Imputation of missing network data. In *Encyclopedia of Social Network Analysis and Mining*; Springer: New York, NY, USA, 2018; pp. 1044–1053.
7. Kim, M.; Leskovec, J. The network completion problem: Inferring missing nodes and edges in networks. In Proceedings of the 2011 SIAM International Conference on Data Mining, Mesa, AZ, USA, 28–30 April 2011; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2011; pp. 47–58.
8. Ouédraogo, F.; Magnien, C. Impact of sources and destinations on the observed properties of the internet topology. *Comput. Commun.* **2011**, *34*, 670–679. [CrossRef]
9. Chung, F.; Garrett, M.; Graham, R.; Shallcross, D. Distance realization problems with applications to internet tomography. *J. Comput. Syst. Sci.* **2001**, *63*, 432–448. [CrossRef]
10. Kannan, S.; Mathieu, C.; Zhou, H. Graph reconstruction and verification. *ACM Trans. Algorithms* **2018**, *14*, 1–30. [CrossRef]
11. Erlebach, T.; Hall, A.; Hoffmann, M.; Mihal'ák, M. Network discovery and verification with distance queries. In Proceedings of the 2006 Conference on Algorithms and Complexity, Rome, Italy, 29–31 May 2006; pp. 69–80.
12. Vasanthakumar, G.U.; Sunithamma, K.; Deepa Shenoy, P.; Venugopal, K.R. An overview on user profiling in online social networks. *Int. J. Appl. Inf. Syst.* **2017**, *11*, 25–42. [CrossRef]
13. Wei, Q.; Hu, G.; Shen, C.; Yin, Y. A fast method for shortest-path cover identification in large complex networks. *Comput. Mater. Contin.* **2020**, *63*, 705–724. [CrossRef]
14. Eriksson, B.; Barford, P.; Crovella, M.; Nowak, R. Learning network structure from passive measurements. In Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, San Diego, CA, USA, 24–26 October 2007; pp. 209–214.
15. Forsati, R.; Barjasteh, I.; Ross, D.; Esfahanian, A.H.; Radha, H. Network completion by leveraging similarity of nodes. *Soc. Netw. Anal. Min.* **2016**, *6*, 1–22. [CrossRef]
16. Tran, C.; Shin, W.-Y.; Spitz, A.; Gertz, M. DeepNC: Deep generative network completion. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**. [CrossRef]
17. Cui, P.; Wang, X.; Pei, J.; Zhu, W. A Survey on Network Embedding. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 833–852. [CrossRef]
18. Bianconia, G.; Pinb, P.; Marsilia, M. Assessing the relevance of node features for network structure. *Proc. Natl. Acad. Sci. USA.* **2009**, *106*, 11433–11438. [CrossRef]
19. Kim, M.; Leskovec, J. Modeling social networks with node attributes using the Multiplicative Attribute Graph model. In Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, 14–17 July 2011; AUAI Press: Arlington, VA, USA, 2011; pp. 400–409.
20. Yang, L.; Kang, Z.; Cao, X.; Jin, D.; Yang, B.; Guo, Y. Topology optimization based graph convolutional network. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 4054–4061. [CrossRef]
21. Newman, M.E.J. The structure and function of complex networks. *SIAM Rev.* **2003**, *45*, 167–256. [CrossRef]
22. Shi, M.; Tang, Y.; Zhu, X. Topology and content co-Alignment graph convolutional learning. *arXiv* **2020**, arXiv:2003.12806.

23. Rafailidis, D.; Crestani, F. Network completion via joint node clustering and similarity learning. In Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), San Francisco, CA, USA, 18–21 August 2016; pp. 63–68.
24. Kaya, M.; Bilge, H.Ş. Deep Metric Learning: A Survey. *Symmetry* **2019**, *11*, 1066. [CrossRef]
25. Zhang, Z.; Cui, P.; Zhu, W. Deep learning on graphs: A Survey. *IEEE Trans. Knowl. Data Eng.* **2020**. [CrossRef]
26. Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
27. Franceschi, L.; Niepert, M.; Pontil, M.; He, X. Learning discrete structures for graph neural networks. In Proceedings of the The 36th International Conference on Machine Learning (PMLR), Long Beach, CA, USA, 9–15 June 2019; pp. 1972–1982.
28. Chen, Y.; Wu, L.; Zaki, M.J. Deep Iterative and Adaptive Learning for Graph Neural Networks. *arXiv* **2019**, arXiv:1912.07832.
29. Yu, D.; Zhang, R.; Jiang, Z.; Wu, Y.; Yang, Y. Graph-Revised Convolutional Network. *arXiv* **2020**, arXiv:1911.07123. [CrossRef]
30. Hao, Y.; Cao, X.; Fang, Y.; Xie, X.; Wang, S. Inductive link prediction for nodes having only attribute information. In Proceedings of the the 29th International Joint Conference on Artificial Intelligence (IJCAI), Yokohama, Japan, 7–15 January 2021; pp. 1209–1215.
31. You, J.; Ying, R.; Ren, X.; Hamilton, W.L.; Leskovec, J. Graphrnn: Generating realistic graphs with deep auto-regressive models. In Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 5708–5717.
32. Alderson, D.; Li, L.; Willinger, W.; Doyle, J.C. Understanding internet topology: Principles, models, and validation. *IEEE/ACM Trans. Netw.* **2005**, *13*, 1205–1218. [CrossRef]
33. Grover, A.; Zweig, A.; Ermon, S. Graphite: Iterative generative modeling of graphs. In Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, CA, USA, 10–15 June 2019; Volume 97, pp. 2434–2444.
34. Xu, D.; Ruan, C.; Motwani, K.; Korpeoglu, E.; Kumar, S.; Achan, K. Generative Graph Convolutional Network for Growing Graphs. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 3167–3171.
35. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR), Palais des Congrès Neptune, Toulon, France, 24–26 April 2017; pp. 1–14.
36. Lin, W.; He, F.; Zhang, F.; Cheng, X.; Cai, H. Initialization for network embedding: A graph partition approach. In Proceedings of the WSDM 2020—The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, 3–7 February 2020; pp. 367–374. [CrossRef]
37. Wang, H.; Leskovec, J. Unifying graph convolutional neural networks and label propagation. *arXiv* **2020**, arXiv:2002.06755.
38. Jia, J.; Benson, A.R. A unifying generative model for graph learning algorithms: Label Propagation, graph Convolutions, and combinations. *arXiv* **2021**, arXiv:2101.07730.
39. Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *arXiv* **2019**. [CrossRef]
40. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
41. Huang, Q.; He, H.; Singh, A.; Lim, S.-N.; Benson, A.R. Combining label propagation and simple models out-performs graph neural networks. *arXiv* **2020**, arXiv:2010.13993.
42. Karrer, B.; Newman, M.E.J. Stochastic blockmodels and community structure in networks. *Phys. Rev. E* **2011**, *83*, 016107. [CrossRef]
43. Abbe, E. Community detection and stochastic block models: Recent developments. *J. Mach. Learn. Res.* **2017**, *18*, 6446–6531.
44. Agarap, A.F.M. Deep learning using rectified linear units (ReLU). *arXiv* **2018**, arXiv:1803.08375.
45. Etworks, N. Geom-gcn: Geometric graph convolutional networks. *arXiv* **2020**, arXiv:2002.05287.
46. Yang, Z.; Cohen, W.W.; Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. In Proceedings of the 33rd International Conference on Machine Learning (ICML), New York, NY, USA, 19–24 June 2016; Volume 48, pp. 40–48.
47. Mernyei, P.; Cangea, C. Wiki-CS: A wikipedia-based benchmark for graph neural networks. *arXiv* **2020**, arXiv:2007.02901.
48. Wei, Q. Network completion via deep metric learning. In Proceedings of the International Conference on Big Data Mining and Information Processing (BDMIP), Qingdao, China, 24–26 July 2020; Volume 1656, p. 012026.
49. Leskovec, J.; Sosič, R. SNAP: A general-purpose network analysis and graph-mining library. *ACM Trans. Intell. Syst. Technol.* **2016**, *8*, 1–20. [CrossRef] [PubMed]