M.Sc. Engg. (CSE) Thesis

Overlapping Community Detection using Deep Graph Neural Networks

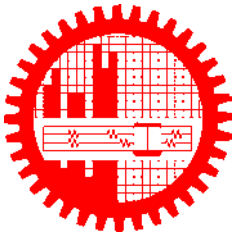
Submitted by

Md Nurul Muttakin

1018052056

Supervised by

Dr. Md Saidur Rahman



Submitted to

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

August 2022

Candidate's Declaration

I, do, hereby, certify that the work presented in this thesis, titled, "Overlapping Community Detection using Deep Graph Neural Networks", is the outcome of the investigation and research carried out by me under the supervision of Dr. Md Saidur Rahman, Professor, Department of CSE, BUET.

I also declare that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Md. Nurul Muttakin

Md Nurul Muttakin

1018052056

The thesis titled "**Overlapping Community Detection using Deep Graph Neural Networks**", submitted by Md Nurul Muttakin, Student ID 1018052056, Session October 2018, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents on August 17, 2022.

Board of Examiners

1. _____

Dr. Md Saidur Rahman — Chairman
Professor — (Supervisor)
Department of CSE, BUET, Dhaka

2. _____

Dr. Mahmuda Naznin — Member
Professor and Head — (Ex-Officio)
Department of CSE, BUET, Dhaka

3. _____

Dr. Md. Monirul Islam — Member
Professor
Department of CSE, BUET, Dhaka

4. _____

Dr. Mohammad Mahfuzul Islam — Member
Professor
Department of CSE, BUET, Dhaka

5. _____

Dr. Chowdhury Mofizur Rahman — Member
Professor — (External)
Department of CSE
United International University
Dhaka

Acknowledgement

Firstly, I would like to show my gratitude to the Almighty for being able to complete my thesis. Then, I am immensely grateful to Professor Dr. Md. Saidur Rahman for giving me intellectual freedom, providing thorough guidance from beginning to the end, and thoughtful supervision at every step of my work. I would like to thank Dr. Iqbal Hossain for his guidance, supports, and suggestions throughout the entire journey of this study and providing us computing support from the University of Arizona to run machine learning experiments. I would like to show my heartfelt gratitude to the honorable board of examiners for thier thoughtful suggestions, comments, and evaluation of my dissertation. Finally, I am thankful to ICT division of Bangladesh Government for their grant on this research.

Dhaka
August 17, 2022

Md Nurul Muttakin
1018052056

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Artificial intelligence (AI) is comparable to human intelligence in many cases. Especially in the case of graph-structured data, there are many AI algorithms and machine learning models which not only can understand these graph-structured data successfully but are also able to predict the future with high precision. Many researchers developed machine learning models to be applied to the graph-structured data. The most successful machine models acting on the graphs are the graph convolutional networks (GCNs). Some research has considered applying GCNs to tackle the overlapping community detection problem. Indeed, overlapping community detection is a key problem in graph-structured data mining. However, the existing GCN-based approach considered a shallow (2-layered) GCN which has an essential limitation in the detection of communities with larger sizes. A viable solution can be the use of deep GCNs. However, training deeper GCNs is notoriously difficult due to several key issues like over-smoothing and vanishing/exploding gradient problems. Some researchers have attempted to resolve these issues in the case of regular graphs generated from point clouds. Unfortunately, it is still challenging how to incorporate deep GCNs in the case of general irregular graphs. In this study, we have resolved the issues of deep GCNs in the case of irregular graphs in an overlapping community detection framework. In particular, we design a deep dynamic residual GCN (DynaResGCN) based on our dynamic dilated aggregation mechanisms in general irregular graphs. Additionally, we introduce weighted dilated aggregation mechanisms to incorporate the information coming from edge-weights of weighted graphs. Finally, we develop a unified end-to-end encoder-decoder based framework to detect overlapping communities in networks. The deep DynaResGCN model is used as the encoder, whereas we incorporate the Bernoulli-Poisson model as the decoder. Consequently, we apply our overlapping community detection framework in a research topics dataset, a set of networks from Facebook, and in a set of very large co-authorship networks. Our experimentation on these datasets shows significantly superior performance over many state-of-the-art methods for the detection of overlapping communities in networks. Finally, this work has many different important applications, ranging from bioscience to social science and any tasks where there is a network structure.

Chapter 1

Introduction

1.1 Overview

Network structured data are ubiquitous. Almost every domain of science and engineering has to deal with networks. The nodes of real-world networks often form special groups where link density is high, and the density of edges is low among these groups. This property of networks is called *community structure* [3]. Human society is composed of communities that are in many cases virtual due to the wide use of online social platforms. Indeed, social communities are of great importance to social scientists and have been studied thoroughly for a long time [4–7]. Communities in the global financial network play a key role to influence the transitions of the global economy and financial system [8]. In protein-protein interaction networks, protein communities act as functional modules to perform specific tasks in cells [9–11]. As a consequence, community detection is an essential tool to analyze different kinds of networks such as social networks, research topics networks, protein-protein interaction networks, ecological networks, financial networks, etc., and to understand their structure.

The communities in most real-world networks, especially social networks, overlap [12]. The same participant participates in different communities at the same time. Therefore, detecting overlapping communities in different networks is of utmost importance, especially in the case of social networks. Every network is naturally represented by a graph. We can consider every participant in the network as a node and every connection as an edge of a graph. As a result, the overlapping community detection problem in different kinds of networks is exactly the same as the overlapping community detection problem in graphs. In the case of graphs, we define a community as a set of nodes having higher connectivity among the nodes of the community than the connectivity from community nodes to outside nodes. In the case of overlapping communities, two different communities have nodes in common. Thus a single node may belong to multiple communities. If the number of communities is given, we can define a community affiliation matrix where columns represent the communities and rows represent the nodes. The value of

each cell is either zero or one. If a node belongs to a community, the value of the corresponding cell would be one. A zero in a cell denotes that the corresponding node does not belong to the corresponding community. In Figure 1.1, we show a simple community affiliation matrix as F. In this study, our aim is to find the community affiliation matrix given a graph. There are many different approaches to attack this problem. For instance, non-negative matrix factorization of the adjacency matrix generates both community embedding and community affiliation matrix. Some recent works [13] consider a graph neural network-based approach to solving this problem. In this thesis, we focus on graph neural networks to generate a community affiliation matrix of a given network/graph. We also resolve several challenges to train deep graph neural networks on general irregular graphs/networks. Finally, we develop an encoder and decoder-based framework for overlapping community detection where the encoder is a deep graph neural network and the decoder is the Bernoulli-Poisson model. We also considered several strategies to incorporate the edge-weight of the weighted graphs. We perform experimentation over many datasets where our developed methods perform significantly better than many state-of-the-art methods.

1.2 Problem Definition

We focus on solving overlapping community detection problems in networks. Every network can be represented as a graph. Thus we have to detect overlapping communities in graphs. A disjoint community detection algorithm assigns every node of a given graph to a single community whereas an overlapping community detection algorithm may assign a node to multiple communities. Indeed, community associations of all the nodes can be represented by a community affiliation matrix F where rows represent nodes and columns represent communities. Each entry F_{ij} contains the information on whether i-th node belongs to the j-th community. This information can be represented by a binary constant where one would indicate community association. In this problem, our input is a graph that may have attributes and the output is the community affiliation matrix. In a more concrete sense, the adjacency matrix A and attribute matrix X would be given as input to generate/construct the affiliation matrix F.

1.3 Motivation and Applications

Overlapping community detection has applications in every area of science and engineering which deals with network structured data. For instance, in the case of social networks, overlapping community detection is essential to understand the functional structure of the network, predict social dynamics, understand how rumor propagation works through overlapping nodes, grasp and control public sentiments, product marketing, identify fraudulent groups of people, etc. In fact, outliers communities often correspond to adversarial activities. In neuroscience, functional

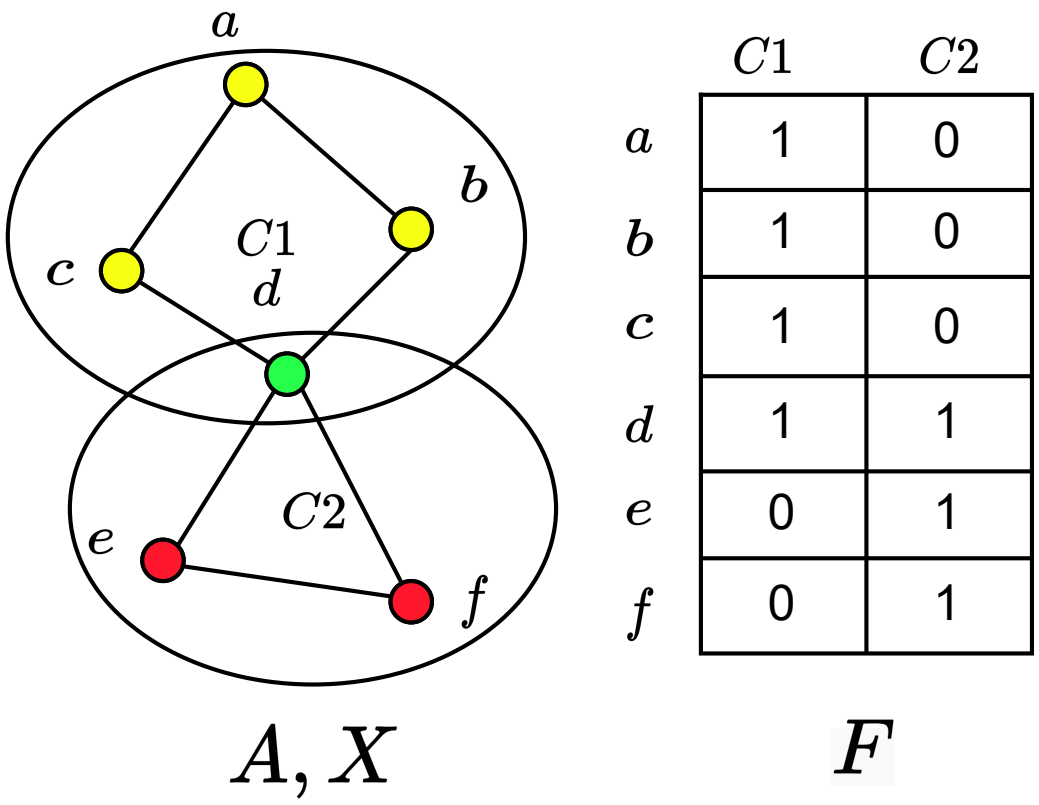


Figure 1.1: Overlapping communities and affiliation matrix.

parts of the brain can be identified through overlapping community detection in biological neural networks [14]. Overlapping community detection gives us the nodes which are involved in more than one community. These overlapped nodes have various applications in different domains. For instance, in the case of social networks, these nodes are crucial for information propagation from one community to another community. In an ecological system/network, these overlapped nodes denote special species which are important for the balance of the ecosystem. In telephone networks, outlier communities can be observed closely as they can be potential suspicious communities. In fact, the security department of a country must observe different communities in different types of networks such as social networks, telephone communication networks, cell phone call networks, etc.

Real life networks or graphs are mostly non-regular. The degree distribution often follows the power-law distribution. For example, the Facebook graphs consist of small number of nodes with very high degrees while a large number of nodes have low degrees. This implies that Facebook graphs are irregular graphs. Other examples include protein-protein interaction networks. Each protein interacts with different number of proteins which eventually makes the protein-protein interaction network irregular. In fact, community detection in regular graphs has no significance because regular graphs have no communities at all. Therefore it is of utmost importance to consider irregular graphs in overlapping community detection setup.

Our work advances deep graph convolutional networks to a new stage where they can handle variable-sized neighborhoods of irregular graphs in a better way. Indeed, graph neural networks have applications in many areas of science and engineering. In particle physics, particle interaction networks can be modeled as graphs. Many researchers [15] considered graph neural networks to predict particle interaction behavior, particle reconstruction dynamics, etc. Therefore, our work would benefit particle physics in a better way. In fact, this is an example application of graph neural networks. In reality, there are thousands of applications of graph neural networks in many different scientific domains. As a consequence, our research has numerous applications and an immense impact on overall science and engineering.

1.4 Literature and Challenges

Many works in the literature are related to non-overlapping community detection [16, 17]. In the case of non-overlapping community detection, node embedding approaches perform well [18, 19]. However, most of these approaches are not well suited for overlapping community detection due to the lack of scalable and reliable clustering methods for high-dimensional vectors. A viable approach to resolve this issue is to generate the node embedding as community affiliation which can be done effectively and efficiently using GCNs [13].

Among the classical approaches for overlapping community detection, label propagation algorithms are prominent [20]. For instance, SLPA is an overlapping community detection

algorithm based on the idea of label propagation [21]. DEMON [22], a node-centric bottom-up overlapping community detection algorithm, also leverages ego network structure with overlapping label propagation. Other algorithmic approaches include seed-set expansion [23, 24], evolutionary algorithms [25], and heuristics-based methods [26–29]. Some heuristics-based works considered game-theoretic concepts [30].

Many studies consider some kinds of matrix factorization along with probabilistic inference [1, 31–36]. One of the earliest works in this regime is BigClam [1]. BigClam was designed for overlapping community detection in large networks [1]. BigClam uses gradient ascent to create an embedding which is later used for determining the community affiliation of nodes. MNMF [37] uses modularity-based regularization with non-negative matrix factorization. DANMF uses a telescopic non-negative matrix factorization approach based on a deep auto-encoder to learn membership distribution over nodes [2]. Some works [38, 39] performed factorization of modularity matrices using neural nets, while other works considered adversarial learning [40, 41] or deep belief networks [42] to learn community affiliation. However, they did not use graphs in their neural network architecture, which is crucial to achieving superior performance [13]. Finally, NOCD [13] used special neural networks called graph neural networks to detect overlapping communities.

Graph neural networks (GNN) are a class of specialized neural networks that can operate on graph-structured data [43–45]. In general, GNNs use a special kind of convolution operation called graph convolution [44]. In graph convolution, each node aggregates information from the neighboring nodes. Before aggregation, the information of neighboring nodes is transformed by a learnable kernel. One of the major goals of graph learning is to generate node embedding to perform downstream induction tasks, such as node classification, edge labeling, clustering, or community detection [46–49]. The authors of NOCD [13] first developed a graph neural network-based approach for overlapping community detection. In that approach, they used a two-layer shallow graph convolutional network as an encoder to generate community embedding and a decoder based on the Bernoulli-Poisson model [1, 31, 50] to reconstruct the original graph from the embedding. However, a shallow GCN has essential limitation to capture communities with a larger community diameter (> 2) as it can aggregate information only up to two hops from a node. *Community diameter* [51] refers to the maximum shortest path distance between any two nodes of a community-induced subgraph (Figure 1.2). One solution to the issue is to consider a deep graph convolutional network (DeepGCN) [52] instead of the shallow GCN. However, Li et al. [52] proposed the DeepGCN in the case of point cloud learning where the considered graph is a regular graph while the real-world networks are mostly irregular graphs (having variable-sized neighborhoods). Thus it is essential to design a DeepGCN which can handle the irregular graphs to detect overlapping communities with a larger diameter.

In fact, training deeper GCNs is challenging because of over-smoothing and vanishing gradient problems [52]. Li et al. [52] overcame this challenge in the case of point cloud learning by

borrowing the idea of residual connection and dilated aggregation from a classical CNN-based architecture called ResNet [53]. *Residual connection* means having direct connectivity from the input of a layer to the output of that layer bypassing the non-linearity in the activation function. In the case of GCNs, each node aggregates information from the neighboring nodes. If this aggregation mechanism aggregates information from some of the distant neighbors while skipping some of the nearest neighbors, it is called *dilated aggregation*. In the case of dilated aggregation, it is possible to incorporate some sort of randomization while considering neighbors, which introduces the concept of *edge dynamicity*. Dilated aggregation with edge dynamicity is called *dynamic dilated aggregation*.

It is still challenging to introduce dynamic dilated aggregation in the case of irregular graphs in order to train deep GCNs.

1.5 Research Aim and Objective

Our research aim consists of three parts. Firstly, we would like to solve the overlapping community detection problem in the real world networks effectively and efficiently. Then we would like to handle the irregular nature of the real world graphs and finally we wish to consider edge weights whenever it is available. Our research objectives are summarized below:

- Detect overlapping communities in real-world networks effectively and efficiently
- Develop a deep graph convolutional network (GCN) which can operate on graphs having variable neighborhood sizes and hence develop an overlapping community detection framework based on deep GCNs.
- Design methodologies to incorporate information coming from edge-weights of weighted graphs in deep GCN in order to improve the effectiveness of overlapping community detection systems.

1.6 Thesis Contribution

In this study, we design dynamic dilated aggregation mechanisms in the case of irregular graphs. We incorporate our dynamic dilated aggregation schemes along with residual connection into the GCN to obtain a deep GCN that is able to learn from any graphs. Eventually, we apply our designed deep GCN to detect overlapping communities in different kinds of networks with different sizes. In the case of overlapping community detection, we adopt an encoder-decoder based approach similar to NOCD. Our encoder is a deep GCN model called DynaResGCN which generates the community embedding and the decoder attempts to reconstruct the original graph.

For evaluation, we consider a large research topics network [54], a set of Facebook datasets [55] having reliable ground truth information, and a set of very large co-authorship networks (nodes $> 10K$) [13] as an additional benchmark. In the case of the topics network, ground truth information related to overlapping communities does not exist. Therefore, we have to rely on unsupervised metrics called fitness functions or quality measures such as conductance, clustering coefficient, density, and coverage [1, 56] to determine the quality of the detected overlapping communities. In addition to the evaluation metrics, we generate heatmaps or colormaps to illustrate community overlap. We compare our results in topics network with existing overlapping community detection approaches, such as NOCD, BigClam, SLPA, DANMF, and DEMON. Our methods show superior performance over most of the considered existing methods in terms of both quality metrics and visualization. In the case of the datasets having ground truth information, we compare with exactly the same baselines as in NOCD considering both node features and without node features in terms of normalized mutual information (NMI) [57]. Our method, DynaResGCN, achieves a clear superior performance over all the baselines in almost every case in terms of NMI. In summary, our contributions are as follows:

- Developed a deep residual GCN (DynaResGCN) model based on novel dynamic dilated aggregation in irregular graphs
- Designed an effective and scalable overlapping community detection framework based on DynaResGCN and Bernoulli-Poisson model
- Detection of overlapping communities in real-world networks outperforming many state-of-the-art methods

1.7 Thesis Organization

The remainder of the thesis is organized as follows. In chapter 2, we introduce preliminary terminologies and some of the prominent related works. Some of the related works considered algorithmic approaches to solve the problem and others considered machine learning based methods. We describe our methodology in chapter 3 where we explain the overlapping community detection model in detail. Indeed, we follow an encoder-decoder based approach to solve the problem. Chapter 4 deals with our experimental study. We also present our results in this chapter. Additionally, the results are analyzed thoroughly in in chapter 4 where we discuss our insights with overall findings. Finally, chapter 5 provides our conclusions with necessary discussions on limitations and future work.

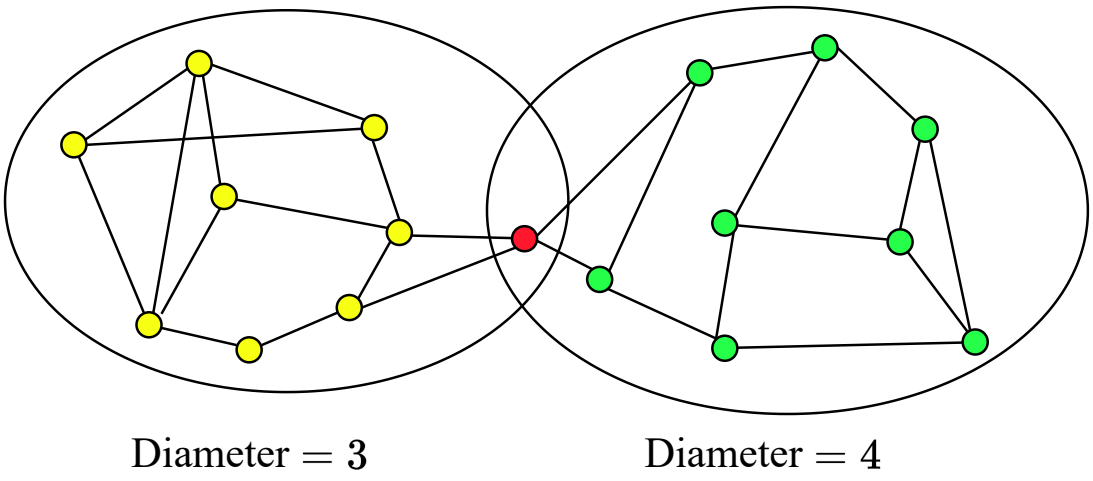


Figure 1.2: Two overlapping communities with community diameter > 2.

Chapter 2

Background and Related Works

In this chapter, we introduce the necessary terminologies that will be used throughout the thesis. We consider an undirected unweighted graph as $G = (V, E)$, where V is the set of vertices and E is the set of edges and weighted graphs as $G = (V, E, W)$. We represent the adjacency matrix as $A \in \{0, 1\}^{n \times n}$, the number of vertices as n, the set of vertices as $V = \{1, \ldots, n\}$, and the set of edges as $E = \{(u, v) \in V \times V : A_{uv} = 1\}$. For each node, we may consider a d-dimensional feature vector. We denote the feature matrix as $X \in \mathbb{R}^{n \times d}$. If there is no feature associated with a node, X is an identity matrix. Let us assume C is the set of communities and the cardinality of C is k. In the case of overlapping community detection, we assign each node to some of the communities in C. However, for each node, there is a strength associated with each different community. We define this strength as the community affiliation of the node. For each node, there is a k-dimensional community affiliation vector. We assign the community affiliation as a non-negative real number. Therefore, we can obtain a community affiliation matrix as $F \in \mathbb{R}_{\geq 0}^{n \times k}$. For a node u, we denote the membership strength of community c as F_{uc}. We use a threshold to obtain a binary community affiliation matrix as $F \in \{0, 1\}^{n \times k}$. Therefore, each node can be assigned to multiple communities. It is also possible that some nodes have no communities at all. The following text first describes the basic concepts related to our thesis, and then some related works.

2.1 Background

Here, we discuss some necessary concepts related to our thesis. We mainly discuss about neural networks, convolutional neural networks, graph neural networks, graph convolutional networks, and related community detection methods.

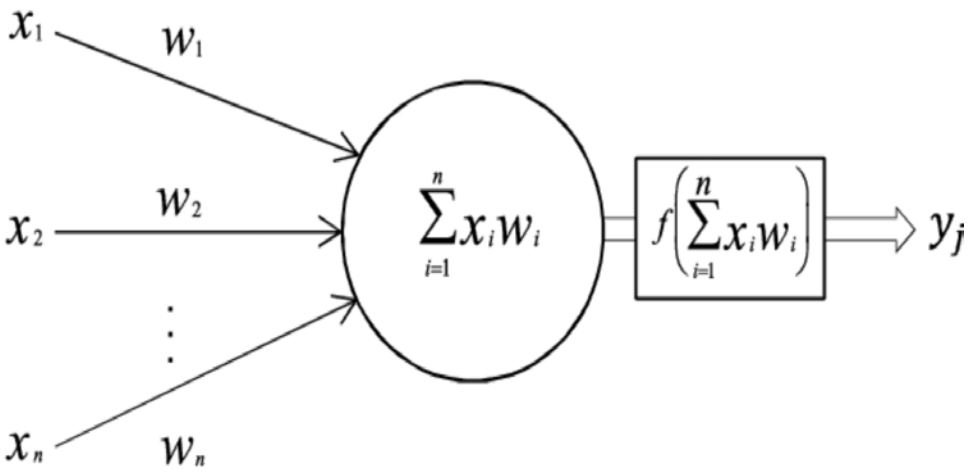


Figure 2.1: Operation of a single artificial neuron

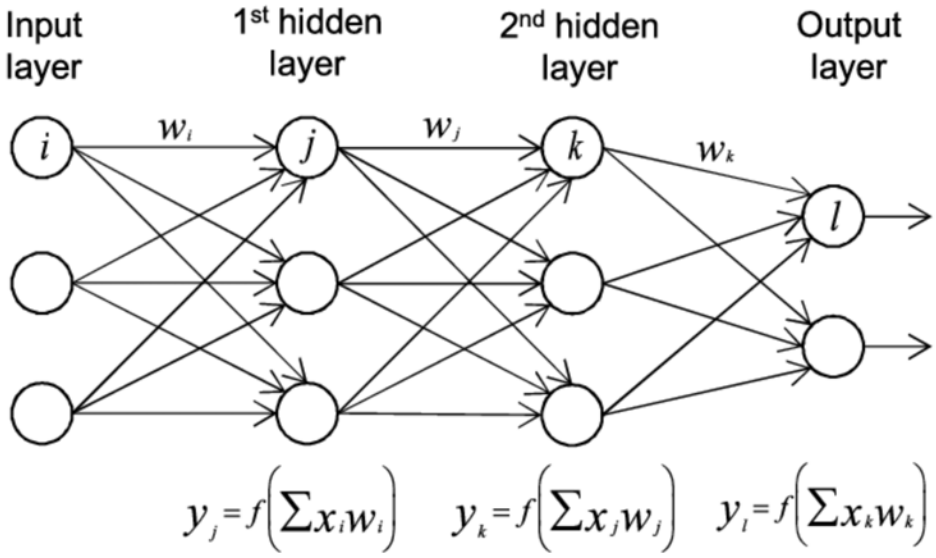


Figure 2.2: A simple neural network where neurons are connected in a computational graph

2.1.1 Neural Network

Artificial neural networks are a class of machine learning models. The smallest computing unit of a neural network is a neuron (Figure 2.1). The input of a neuron is a vector. A neuron is like a parameterized function whose output varies with different parameters for the same input. These neurons connected in a computational graph are called a neural network (Figure 2.2). For more information, readers may study the overview article by Schmidhuber [58].

2.1.2 Convolutional Neural Network

A convolutional neural network (CNN) is a class of neural network based on convolution operation (Figure 2.3). A convolution operation is a process that filters different important features from the given image. A convolutional neural network consists of multiple convolutional layers. Each convolutional layer extracts specific features. In fact, deeper layers capture more high level features. For example, if build a convolutional neural network for animal classification, the deeper layers will capture features ear, eyes, nose, etc., while the initial layers may capture low level features which would make the high level features finally. There are a lot of online resources to understand convolution in detail including this [59]. Earlier works required hand-designed feature engineering. However, CNNs do feature engineering inherently. We apply CNN on image structured data. It is possible to consider an image as a regular graph, where each pixel is a node and each pixel is connected to eight neighboring pixels. Thus, we may consider an image as an 8-regular graph. Therefore, we may use a CNN for a regular graph. However, these classical CNNs are not able to process general graphs. Special treatment is required to perform convolution on general graphs.

2.1.3 Graph Neural Network

Graph Neural Networks (GNN) is designed to handle the inability of classical Neural Networks (NN) to process general graphs. Scarselli *et. al.* proposed the graph neural network model [60] to alleviate the limitations of classical neural networks to process graph-structured data. According to this model, a label vector and a feature vector are associated with each node of a graph. This model predicts the class of the graph or the classes of the nodes of the graph and provided the labels and feature vectors for all nodes. There are two types of function associated with this model: 1) transfer function and 2) output function. The transfer function transforms the feature vector of each node to a new transformed vector based on the features of neighboring nodes. The transfer function is applied until the node features come to a stable state, i.e., the features do not change anymore. Finally, an output function is applied to obtain the predicted class of each node. The output function can be node-wise or entire graph-wise. The transfer function and

output function are realized using a classical neural network. This is the basic idea of the very first graph neural network model.

2.1.4 Graph Convolution Networks

Graph Convolution Networks (GCN) are GNNs that can perform convolution operations in general graphs (Figure 2.4). In the case of GCNs, each node aggregates the feature vectors of its neighboring nodes to achieve a richer feature vector. For each vertex u, a d-dimensional feature vector is considered as $h_u \in \mathbb{R}^d$. The whole graph G can be represented by concatenating the feature vectors of all the nodes as $h_G = [h_{u_1}, h_{u_2}, \ldots, h_{u_n}]^T \in \mathbb{R}^{n \times d}$. A general graph convolution operation is defined based on two operations: an aggregation operation and an update operation.

$$G_{l+1} = Update(Aggregate(G_l, W_l^{agg}), W_l^{update}). \tag{2.1}$$

G_l is the input graph with features at the l-th layer and G_{l+1} is the output graph with updated features at the $(l+1)$-th layer. For the aggregation operation, the learnable weights matrix is W^{agg} and for the update operation, the learnable weights matrix is W^{update}. In most graph neural networks, the aggregation function aggregates the features from the neighbors and the update function updates the feature vector of a node based on the output of the aggregation function. In most cases, the update function is a non-linear function. Many variations of these functions are also possible. The mean aggregation function [44], max-pooling aggregation function [43, 61, 62], attention aggregation function [63], LSTM aggregation function [64], etc., are some of the existing aggregation functions. MLP (multi-layer perceptrons) [43, 65], gated network [66], etc., are some of the variations of update functions.

2.1.5 Shallow GCN Model

GCNs have very low numbers of layers. In previous work, NOCD [13] used a two-layered graph convolution network that was first proposed by Kipf and Welling [44]. We refer to this two-layer GCN as a shallow GCN. According to NOCD [13], the GCN model is defined as follows:

$$F := GCN_\theta(A, X) = ReLU(\hat{A}ReLU(\hat{A}XW^{(1)})W^{(2)}) \tag{2.2}$$

Here, $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ is the normalized adjacency matrix, and $\tilde{A} = A + I_N$ is the adjacency matrix considering self-loops. The diagonal degree matrix of $\tilde{A}$ is $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Here, the $ReLU$ function works as a non-linear update function, and aggregation is done through a weighted sum of the features of neighbors. This weight is a learnable parameter. According to this model, deep architectures do not work well [13, 44]. Some modifications to this shallow

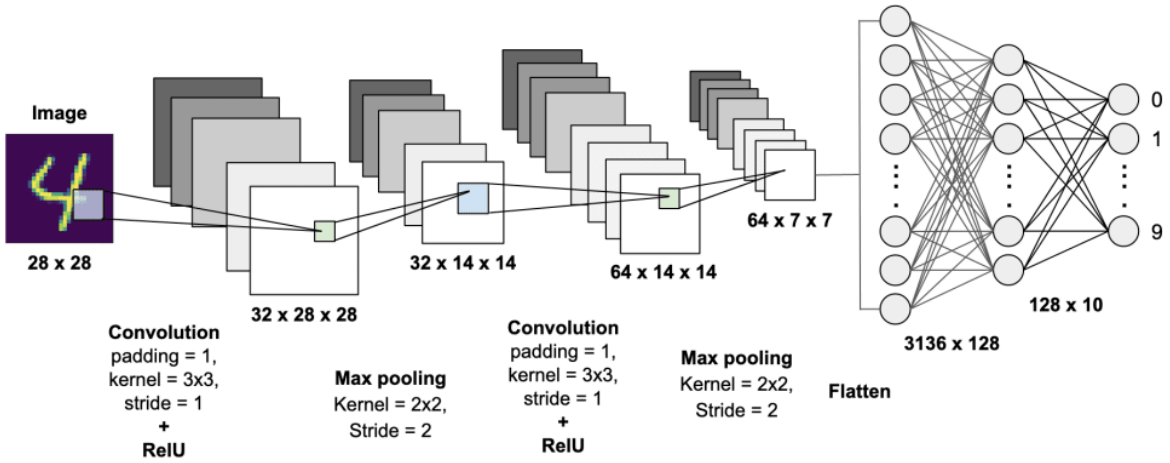


Figure 2.3: A CNN to predict image classification

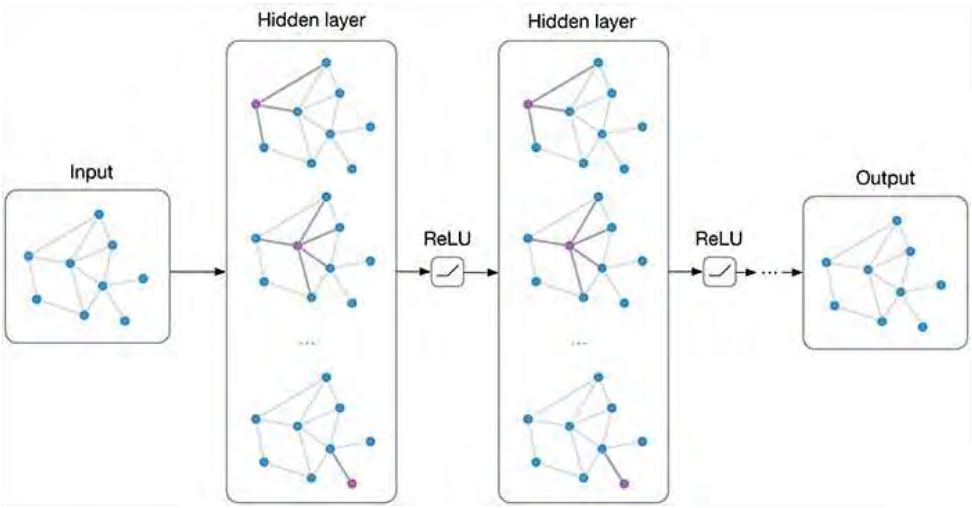


Figure 2.4: A high-level architecture of shallow graph convolutional networks with 2 hidden layers [44]

model were introduced [13]. These are: (1) batch normalization after the first GCN layer and (2) L_2 regularization applied to all weight matrices. In fact, shallow GCNs have some limitations. For instance, the number of layers is limited. When the number of layers is increased, a GCN faces the over-smoothing and the vanishing gradient problem.

2.2 Overlapping Community Detection Methods

In the following, we include most of the prominent related works. More specifically, we discuss speaker-listener label propagation algorithm, cluster affiliation model for big networks, neural overlapping community detection model, etc.

2.2.1 Speaker-listener Label Propagation Algorithm

The Speaker-listener Label Propagation Algorithm (SLPA) [21] is a general framework to analyze overlapping communities in social networks. According to this algorithm, nodes exchange information (label) based on dynamic interaction rules. This framework is designed to analyze both individual overlapping nodes and also the whole community. SLPA is an extension of the previous label propagation algorithm (LPA) [67]. Each node in the LPA has a single label. This label is iteratively updated by the majority of labels in the neighborhood. After completion of the algorithm, non-overlapping (disjoint) communities are discovered. To allow overlap, each node is allowed to have multiple labels. There are two issues related to this algorithm: 1) disseminating node information efficiently and, 2) aggregating/processing information received from neighbors in an information preserving way. Another critical issue is related to information maintenance. To resolve all of these issues and manage data at different nodes, the authors of SLPA designed a speaker-listener-based algorithm to resolve all of these issues and manage data at different nodes. In the SLPA, each node can act as a listener or a speaker, depending on whether it is absorbing information or disseminating information. The algorithm consists of three stages. First, each node label is initialized with a unique value. Then, an iterative process continues until a stopping condition is met. The process is as follows [67]:

- A node is considered a listener
- Each neighboring node sends a label to the node, according to some predefined *speaking* rule
- The listener accepts one label from all of the labels received, according to a predefined *listening rule*

Finally, post-processing of the accumulated labels for each node determines the communities. The algorithm runs in $\mathcal{O}(Tn)$. Here, n is the number of nodes in the graph and T is the maximum number of iterations.

2.2.2 Cluster Affiliation Model for Big Networks

BigClam (Cluster Affiliation Model for Big Networks) was originally designed for large networks [1]. It is a probabilistic generative model for graphs to capture network community structure based on the community distribution of each node. The whole idea of BigClam is based on a bipartite community affiliation network. The nodes of a network (graph) form one partite set, and the communities form another partite set. The links between the nodes and the communities represent the membership of a node in a community. This link or community affiliation edge has a non-negative weight representing the membership strength of a node to a community. The generative model is based on the fact that the higher number of communities two nodes share, the greater the probability of connecting these two nodes with an edge. Let us assume a graph $G(V, E)$, the corresponding set of communities C, and the set of affiliation edges as M. Then the bipartite affiliation network is represented as $B(V, C, M)$. BigClam is a generative model, where the edges of the graph can be generated from the affiliation network. The bipartite affiliation network $B(V, C, M)$ is given and we have to generate a graph $G(V, E)$. The vertices V of G are already specified in the affiliation network B. We need to generate the E for G. For each node $u \in V$ and community $c \in C$, there is a non-negative weight F_{uc}. When there is no affiliation edge between u and c, then $F_{uc} = 0$. If two nodes u, v belong to the same community c, then this community c will connect these two nodes u, v with probability $1 - exp(-F_{uc}.F_{vc})$. This is the edge probability for one community. The overall edge probability between u and v, considering all communities, is $1 - exp(-\sum_c F_{uc}.F_{vc})$. If F is the non-negative matrix where F_{uc} is the affiliation weight between node u and community c, then the edge probability $p(u, v)$ between nodes $u, v \in V$:

$$p(u, v) = 1 - exp(-F_u.F_v^T) \tag{2.3}$$

Here, the weight vector of node u is F_u.

Community detection using the BigClam model is the reverse problem of generating the graph $G(V, E)$ given the model $B(V, C, M)$ and the community affiliation matrix F. In community detection, the community affiliation matrix F is determined by the underlying graph $G(V, E)$. The number of communities k is given, and the BigClam model finds the most likely affiliation matrix $\hat{F}$ maximizing log-likelihood $l(F) = \log P(G|F)$.

$$\hat{F} = \arg\max_{F \geq 0} l(F) \tag{2.4}$$

$$l(F) = \sum_{(u,v)\in E} \log\left(1 - exp(-F_u F_v^T)\right) - \sum_{(u,v)\notin E} F_u F_v^T. \tag{2.5}$$

This optimization is similar to the non-negative matrix factorization approach (NMF) [68]. The goal of the optimization is to learn the affiliation matrix F that best approximates the underlying graph $G(V, E)$. Existing NMF methods [68, 69] have been modified by BigClam for large networks [1]. BigClam a block coordinate ascent algorithm [69, 70] to solve the optimization problem. After finally obtaining the community affiliation matrix F, a threshold δ is used to determine the community affiliation of each node. If $F_{uc} < \delta$, then node u does not belong to the community c.

To solve the optimization problem in (2.5), the authors of [1] considered a block coordinate ascent algorithm. In this approach, they update for one node while keeping fixed for all the other nodes. This makes the problem a convex optimization problem for each node u. As consequence, for each node u the following subproblem is solved:

$$\arg\max_{F_{uc}\geq 0} l(F_u), \tag{2.6}$$

where

$$l(F_u) = \sum_{v\in\mathcal{N}(u)} \log(1 - \exp\left(-F_u F_v^T\right)) - \sum_{v\not\in\mathcal{N}(u)} F_u F_v^T \tag{2.7}$$

where $\mathcal{N}(u)$ is the set of neighbors for node u. This convex optimization problem is solved using projected gradient ascent as follows:

$$\nabla l(F_u) = \sum_{v\in\mathcal{N}(u)} F_v \frac{\exp\left(-F_u F_v^T\right)}{1 - \exp\left(-F_u F_v^T\right)} - \sum_{v\notin\mathcal{N}(u)} F_v \tag{2.8}$$

The step size is computed using a backtracking line search. Moreover, F_u is projected into a non-negative vector space using $F_{uc} = \max(F_{uc}, 0)$. In the case of large networks, this approach is not scalable

2.2.3 Democratic Estimate of the Modular Organization of a Network

DEMON stands for Democratic Estimate of the Modular Organization of a Network [22], an algorithm designed to discover communities in complex networks. In this approach, each node gives community labels to its neighboring nodes. Thus, every node obtains community labels from its neighboring nodes. This approach is called a *democratic approach* because each node can vote for every neighboring node. Then, the communities are discovered from

the labels for each node. Since one node can have multiple labels, one node can belong to multiple communities. After obtaining labels from the neighborhood, the communities can be discovered using different merging algorithms. For each node, its corresponding ego network is extracted. Then, a label propagation community detection algorithm [67] is applied to remove the corresponding node. Finally, a combination step is executed to obtain the communities.

2.2.4 Modularized Non-negative Matrix Factorization

Wang et al. [37] proposed the Modularized Nonnegative Matrix Factorization (M-NMF) model. In this approach, the authors designed a community-preserving matrix factorization scheme. The modularity-based community detection model and NMF-based representation learning model are jointly optimized in a unified learning framework. Basic intuition indicates the nodes belonging to the same community should have similar representations. The proposed M-NMF model is optimized to preserve both pairwise node similarity and community structure. For pairwise node similarity, the first-order and second-order proximities of nodes are incorporated to learn representations using matrix factorization. A modularity constraint term is incorporated to detect communities. In the following text, we describe their model.

Modeling Community Structure

The M-NMF model considers an undirected graph $G = (V, E)$. The number of nodes is considered as n and the number of edges is considered as e. The graph G is represented as $\mathbf{A} = [A_{ij}] \in \mathbb{R}^{n \times n}$. The goal is to learn embedding of nodes as $U \in \mathbb{R}^{n \times m}$ where $m(m \leq n)$ is the length of the embedding vector for a single node. In this work, a modularity maximization-based approach was incorporated in order to detect overlapping communities. For a network with two communities modularity is defined as follows [37]:

$$Q = \frac{1}{4e} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2e}) h_i h_j \tag{2.9}$$

where k_i is the degree of node i and $h_i = 1$ if node i belong to the first community and $h_i = -1$ if node i belong to the second community. The expected number of edges between node i and node j is $\frac{k_i k_j}{2e}$ when edges are placed at random. In fact, modularity is a metric that measures the difference between the number of edges in the communities and the expected number of edges in an equivalent network where edges are placed at random. A modularity matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ is defined with element

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2e}. \tag{2.10}$$

Now Q becomes as follows:

$$Q = \frac{1}{4e} h^T B h \tag{2.11}$$

where $h = [h_i] \in \mathbb{R}^n$ is the community membership vector for all the nodes.

It is possible to extend this notation to more than 2 communities. Let us assume we have $k > 2$ communities. It is possible to generalize this modularity metric for more than 2 communities with a community affiliation matrix $H \in \mathbb{R}^{n \times k}$ where in each row of H only one element is 1 and all other elements are zero. Thus the formulation becomes as follows:

$$Q = tr(H^T B H), \tag{2.12}$$

$$\text{s.t.} \quad tr(H^T H) = n \tag{2.13}$$

where $tr(H^T H)$ is the trace of the matrix $H^T H$.

Modeling Microscopic Structure

To model microscopic structure it is required to define first-order proximity and second-order proximity. This work considered the adjacency matrix as the first-order proximity. The first-order proximity is the most obvious structural representation of a network. It is important to maintain first-order proximity when embedding vectors for different nodes are generated. Indeed, if two nodes are connected by an edge in the network then it is plausible to have similar embedding vectors for these two nodes. However, it does not imply that if two nodes do not share any edge then these nodes are dissimilar. In many cases, two nodes share many common neighbors even if they are not directly connected. In such a case, these two nodes having many common neighbors should have similar embeddings. Let us assume S^1 is the first-order proximity matrix. Then second-order proximity matrix is defined as S^2 and we also assume $\mathcal{N}_i = (S^1_{i,1}, \ldots, S^1_{i,n})$ is the first order proximity vector for node i where $S^1_{i,j}$ defines the first order proximity of node i with node j. The cosine similarity of vector $\mathcal{N}_i$ and $\mathcal{N}_j$ is used to define the second-order proximity of node i and node j. Thus second-order proximity between nodes i and nodes j is defined as follows:

$$S^2_{ij} = \frac{\mathcal{N}_i \mathcal{N}_j}{\|\mathcal{N}_i\| \|\mathcal{N}_j\|}, \tag{2.14}$$

where $\|\mathcal{N}_i\|$ is the euclidean norm of the vector $\mathcal{N}_i$ and $\|\mathcal{N}_j\|$ is the euclidean norm of the vector $\mathcal{N}_j$. We know that the second-order proximity should be between 0 and 1 as it is the cosine distance between two vectors. The overall similarity matrix is obtained as follows:

$$S = S^1 + \eta S^2 \tag{2.15}$$

where $\eta > 0$ is the weight for the second-order proximity. The authors [37] set the value of $\eta = 5$. Now it is possible to introduce a framework similar to non-negative matrix factorization (NMF). In this framework, we would like to factorize the matrix S into two non-negative matrices where one matrix is called the basis matrix, and another matrix is called the representation matrix. The basis matrix is represented by $M \in \mathbb{R}^{n \times m}$ and the representation matrix is denoted by $U \in \mathbb{R}^{n \times m}$ where m is the length of the representation vector for a single node. Indeed, the i-th row of the matrix U denotes the representation vector or embedding for node i. With these two matrices U and M, the goal is to approximate the similarity matrix S, which makes the objective function as follows:

$$\min \|S - MU^T\|_F^2 \tag{2.16}$$

$$\text{s.t. } M \geq 0,\ U \geq 0 \tag{2.17}$$

It is also possible to add higher-order proximity in this model. In fact, in our future work, we plan to add this similarity as a loss function to train the GCN encoder. Currently, we are optimizing Bernoulli-Poisson decoder-based loss function which assumes the affiliation matrix based on only first-order proximity. It would be interesting to consider a loss function considering second-order proximity too. Indeed, we are currently planning to consider second-order proximity-based loss optimization which is similar to the triangular decoder method in some other related works.

Unified Network Embedding Model

Now the authors combine the modularity maximization model and similarity maximization model in a single framework. Then they introduce a non-negative community representation matrix. This community representation matrix is denoted as $C \in \mathbb{R}^{k \times m}$. Each row of this matrix C represents a community. For example, the i-th row denotes the representation vector of the i-th community. In fact, the affiliation of node i to community r can be formulated as $U_i C_r$. In fact, the goal is that the community affiliation matrix H should be very close to UC^T. Therefore, the overall objective function is as follows:

$$\min_{M,U,H,C} \|S - MU^T\|_F^2 + \alpha \|H - UC^T\|_F^2 - \beta tr(H^T B H) \tag{2.18}$$

$$\text{s.t. }, M \geq 0, U \geq 0, H \geq 0, C \geq 0, tr(H^T H) = n \tag{2.19}$$

In our work, we will incorporate this type of loss function in our future work. We are mostly interested to consider the second-order proximity part in our current loss function. The intuition is that if two nodes share many nodes in common, then there is a very high probability that these two nodes will belong to the same community, and also the chance of having an edge between these two nodes is also very high. Therefore, it is very important to consider this type of second-order proximity score in our future work.

2.2.5 Deep Auto-encoder like Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is a prominent approach for community detection [1, 34, 36]. NMF approaches have better interpretability and are a natural fit for both overlapping community detection and disjoint (non-overlapping) community detection. DANMF [2] (Deep Auto-encoder like Non-negative Matrix Factorization) is a variant of NMF. NMF-based approaches factorize the adjacency matrix A into two non-negative matrices U, V, where $A \sim UV(U \geq 0, V \geq 0)$. Normally, the columns of V represent the community strength of the nodes of the network. NMF methods typically follow shallow methods, i.e., the adjacency matrix is factorized in one level. The authors of DANMF proposed a hierarchical matrix factorization scheme. Inspired by deep auto-encoders, the adjacency matrix is factorized into multiple levels. DANMF is like auto-encoders where there is an encoder component and a decoder component. Both components have deep structures. The encoder part transforms the original adjacency matrix into a community affiliation matrix and the decoder part reconstructs the original adjacency matrix. Unlike deep-autoencoders, the loss function in DANMF is unified with both the encoder and decoder. Thus, DANMF is similar to deep auto-encoders for representation learning [71].

Deep Non-negative Matrix Factorization

Using Non-negative Matrix Factorization (NMF) we can learn a one-layer mapping U and community affiliation matrix V. This is explained in Figure 2.6. However, real-world networks are complex and have variational structures and hierarchical relationships. Therefore, a single layer of matrix factorization is not sufficient to capture the complex attributes of real-world networks. Perhaps, there are multiple levels of abstractions. As a result, it is natural to consider deep learning-based approaches to learn multiple levels of abstractions. Indeed, the authors of [2] proposed to factorize the mapping matrix U multiple times to capture different structures and hierarchies of the hidden representation. In a more concrete sense, it is possible to factor the adjacency matrix into $p + 1$ non-negative matrices. The scheme is described through the product of factor matrices as follows:

$$A \approx U_1 U_2 \dots U_p V_p, \tag{2.20}$$

where $V_p \in \mathbb{R}_+^{k \times n}$, $U_i \in \mathbb{R}_+^{r_{i-1} \times r_i}$ $1 \leq i \leq p$, and we get $n = r_0 \geq r_1 \geq \dots \geq r_{p-1} \geq r_p = k$. This hierarchical formulation allows multiple level of abstractions which can be achieved as

follows:

$$V_{p-1} \approx U_p V_p, \tag{2.21}$$

$$\vdots \tag{2.22}$$

$$V_2 \approx U_3 \dots U_p V_p, \tag{2.23}$$

$$V_1 \approx U_2 \dots U_p V_p. \tag{2.24}$$

Moreover, we consider non-negativity constraints for V_i $(1 \leq i < p)$. As a consequence, each V_i captures the similarity between nodes with different granularity. Indeed, these abstractions can capture first-order proximity, second-order proximity, third-order proximity, etc., and structural similarity and community-level similarity. It is possible to learn the factor matrices by optimizing the following objective function:

$$\min_{U_i, V_p} \mathcal{L}_D = \|A - U_1 U_2 \dots U_p V_p\|_F^2 \tag{2.25}$$

$$\text{s.t. } V_p \geq 0, U_i \geq 0, \forall i = 1, 2, \dots, p. \tag{2.26}$$

After optimizing this equation it is possible to achieve hidden attributes V_i $(i < p)$ by solving:

$$\|A - U_1 U_2 \dots U_i V_i\|_F^2 \tag{2.27}$$

Deep Autoencoder-like NMF

It is clearly evident that the previous Deep NMF formulation is similar to the decoder component of an autoencoder framework. Therefore, it is essential to consider the encoder component in the NMF-based community detection system. The formulation is straightforward. It should be possible to reconstruct the original adjacency matrix from the community affiliation matrix V with a small error if the community affiliation matrix is almost perfect and ideal. Moreover, it is possible to obtain the community affiliation matrix V very easily from the mapping matrix U and the adjacency matrix A. The idea is that we can project the adjacency matrix in the community affiliation space through the mapping matrix U as follows:

$$V = U^T A \tag{2.28}$$

In fact, if the encoder component and decoder component are unified through a unified loss function, then each component can guide the other through a whole learning process. This strategy may help to obtain the ideal community affiliation matrix of a given network. Therefore, the authors of [2] proposed an objective function to be optimized for the encoder component.

This objective function of the encoder component is described as follows:

$$\min_{U_i,V_p} \mathcal{L}_E = \|V_p - U_p^T \dots U_2^T U_1^T A\|_F^2 \tag{2.29}$$

$$\text{s.t. } V_p \geq 0, U_i \geq 0, \forall i = 1, 2, \dots, p. \tag{2.30}$$

Finally they proposed a unified objective function for deep autoencoder-like Non-negative Matrix Factorization (DeepANMF). This unified objective function is described as follows:

$$\min_{U_i,V_p} \mathcal{L} = \mathcal{L}_D + \mathcal{L}_E + \lambda \mathcal{L}_{reg} \tag{2.31}$$

$$= \|A - U_1 U_2 \dots U_p V_p\|_F^2 + \|V_p - U_p^T \dots U_2^T U_1^T A\|_F^2 + \lambda tr(V_p L V_p^T), \tag{2.32}$$

$$\text{s.t. } V_p \geq 0, U_i \geq 0, \forall i = 1, 2, \dots, p. \tag{2.33}$$

where the graph regularizer $\mathcal{L}_{reg} = tr(V_p L V_p^T)$ is incorporated to introduce intrinsic node pairs properties which are mainly geometric structures and λ is the regularization parameter, L is the Laplacian matrix corresponding to the network. In fact, there are many ways to define L [2]. In this DANMF work, the graph Laplacian matrix is defined as follows:

$$L = D - A \tag{2.34}$$

where D is a diagonal matrix which is defined as follows:

$$D_{ii} = \sum_{j=1}^{n} A_{ij} \tag{2.35}$$

In this D matrix all the entries are zeros except the diagonal entries.

Optimization

The authors of [2] considered an optimization based on pre-training. They train each layer to have good initial factor matrices U_i and V_i. This pretraining approach helps to reduce the overall training time. Indeed, pre-training has been proved effective in many areas of deep learning considering transformer networks for natural language processing, evoformer for predicting 3D structures of proteins which is done by DeepMind (Google). The authors of [2] performed pre-training to reduce training time. They firstly, decompose the adjacency matrix as follows:

$$A \approx U_1 V_1 \tag{2.36}$$

To decompose the adjacency matrix they minimize the following function,

$$\|A - U_1 V_1\|_F^2 + \|V_1 - U_1^T A\|_F^2 \tag{2.37}$$

where $U_1 \in \mathbb{R}_+^{n \times r_1}$ and $V_1 \in \mathbb{R}_+^{r_1 \times n}$. Then they decompose the matrix V_1 as follows:

$$V_1 \approx U_2 V_2 \tag{2.38}$$

To do so, they minimize the following function:

$$\|V_1 - U_2 V_2\|_F^2 + \|V_2 - U_2^T V_1\|_F^2 \tag{2.39}$$

where $U_2 \in \mathbb{R}_+^{r_1 \times r_2}$ and $V_2 \in \mathbb{R}_+^{r_2 \times n}$. After all the layers are pre-trained, they propose fine-tuning by alternating optimization of the considered overall final loss function.

Time Complexity

DANMF is composed of two stages as follows:

1. Initial pre-training process, and
2. Final fine-tuning process.

The computational complexity of the pre-training process is

$$\mathcal{O}(pt_p(n^2 r + nr^2)) \tag{2.40}$$

where p is the number of layers and t_p is the number of layers to achieve convergence and r is the maximal layer size among all the layers. The computational complexity of the fine-tuning stage is in the order,

$$\mathcal{O}(pt_f(n^2 r + nr^2 + r^3)) \tag{2.41}$$

where t_f is the number of iterations in the fine tuning process. In general $r < n$. Thus overall complexity is as follows [2]:

$$\mathcal{O}(p(t_p + t_f)(n^2 + nr^2)) \tag{2.42}$$

2.2.6 Neural Overlapping Community Detection

NOCD stands for neural overlapping community detection [13]. The model consists of an encoder and a decoder. The encoder is a shallow graph convolution network (GCN) with two

layers. The decoder is based on the Bernoulli-Poisson model [1]. The output of the encoder is a community affiliation matrix. The decoder attempts to reconstruct the original graph. Finally, a loss is generated based on the reconstruction error. This loss is used to train the GCN model. We use the same framework in this study.

Bernoulli–Poisson model

The Bernoulli–Poisson (BP) model was originally proposed by the authors of [1] is a graph generative model. This graph generative model can be used to detect overlapping communities. This model is somewhat similar to NMF-type models. The Bernoulli-Poisson model is used to generate graphs using the following rules. If the community affiliation matrix is given as $F \in \mathbb{R}_{\geq 0}^{N \times C}$, then the adjacency matrix A_{uv} can be generated using the following distribution:

$$A_{uv} \sim \text{Bernoulli}(1 - \exp(-F_u F_v^T)) \tag{2.43}$$

where F_u is the community affiliation matrix of node u and F_v is the community affiliation vector of node v. Indeed, the higher number of communities nodes u and v have in common, the higher the probability to have an edge between these nodes. In a more concrete sense, the higher the value of the dot product $F_u F_v^T$ is, the more likely to have an edge between the nodes u and v. In other words, the probability of an edge between two nodes u and v is positively correlated with the value $F_u F_v^T$.

This BP model is interesting as it can be used to model nested communities. Moreover, we can also model hierarchical communities using this model [13]. In fact, this model may lead to dense overlaps between communities [12] and it is computationally efficient and scalable [13]. The authors of BigClam proposed to perform community detection in the Bernoulli-Poisson model using maximum likelihood estimation with the coordinate ascent. Some other authors proposed Markov chain Monte Carlo estimation or block coordinate ascent.

Model definition

The authors of NOCD [13] propose to optimize a graph neural network instead of considering a free variable optimization over the community affiliation matrix F. The main idea is to generate a community affiliation matrix given the graph neural networks (GNN):

$$F := \text{GNN}_\theta(A, X) \tag{2.44}$$

It is important to ensure the non-negativity of the affiliation matrix. A rectified linear unit in a neural network can perform this operation. Therefore, a ReLU non-linearity is applied element-wise to the output to make the non-negativity of the community affiliation matrix F.

The negative log-likelihood of the Bernoulli–Poisson model is

$$-\log p(A|F) = -\sum_{(u,v)\in E} \log(1 - \exp(-F_u F_v^T)) + \sum_{(u,v)\notin E} F_u F_v^T \tag{2.45}$$

Due to the sparsity of many real-life graphs, non-edges are much more common than edges. To overcome this issue, the authors of NOCD [13] proposed balancing the terms of the negative log-likelihood according to the standard technique for imbalanced data [72].

$$\mathcal{L}(F) = -\mathbb{E}_{(u,v)\sim P_E}[\log(1 - exp(-F_u F_v^T))] + \mathbb{E}_{(u,v)\sim P_N}[F_u F_v^T] \tag{2.46}$$

Here, uniform distributions over edges and non-edges are represented by P_E and P_N respectively. In our case, we follow the same approach as in NOCD [13]. To learn the community affiliation matrix F, the authors of BigClam [1, 73] directly optimized the affiliation matrix F. Unlike these traditional approaches, we optimize the parameters of the GNN to obtain a more accurate affiliation matrix. The equation (3.9) generates the loss. This loss is back-propagated in the GNN encoder to learn the parameters W_l.

It has several advantages to use graph neural networks for community detection. First, due to the nature of the aggregation scheme in a graph neural network, the graph neural network generates similar community embedding for neighboring nodes. As a consequence, the quality of predictions is improved compared to simpler models. Moreover, these settings help us to incorporate node attributes into the model without any extra formulation. If the node features X are not found, we can simply consider A as node features [44].

Scalability

The Bernoulli-Poisson model may allow us efficiently compute the loss $L(F)$ and its gradients w.r.t. F. Combining the catching trick [1] with this NOCD model, it is possible to reduce the computational complexity from $O(N^2)$ to $O(N+M)$. It is also possible to consider mini-batches of edges and non-edges from a graph when computing the loss function. This will help us to get rid of computing loss for the whole adjacency matrix. Now if the mini-batch size is S, then we can compute the gradient of the loss ∇L in $O(S)$. thus approximately computing ∇L in $O(S)$.

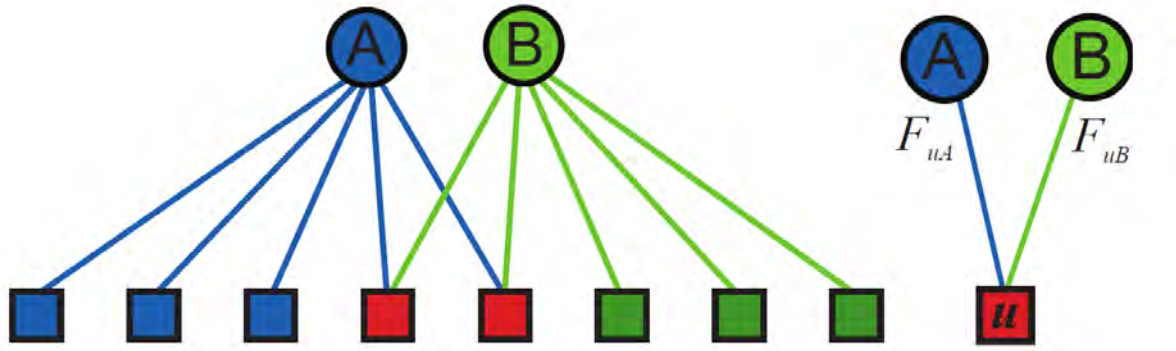


Figure 2.5: BigClam community affiliation network [1].

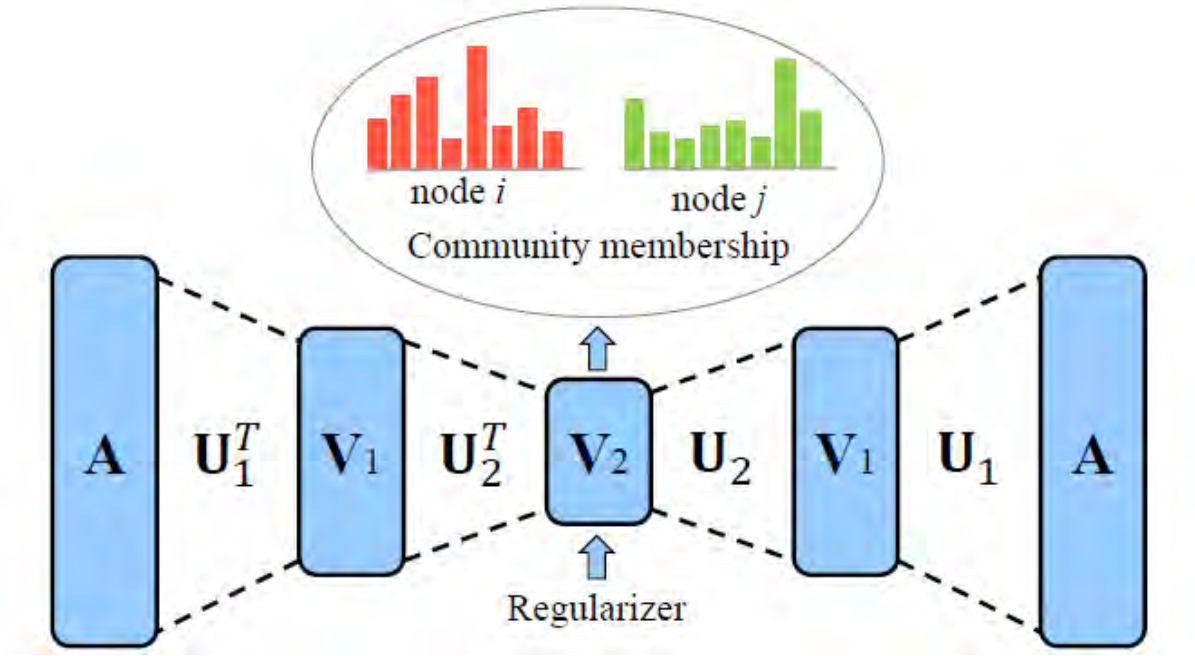


Figure 2.6: NMF and deep NMF [2])

Chapter 3

Overlapping Community Detection Model

Communities are defined on a graph as the dense sub-graphs of the graph. According to the most recent research, a community is a group of nodes. These nodes have a higher probability of forming edges within the group than forming edges with nodes outside of the group [13, 74]. We adopt this definition of a community to develop our framework of overlapping community detection. We consider a graph generative model $p(G|F)$, for a graph G [1, 13]. Here, F is the community affiliation matrix. This is a $N \times C$ matrix, where N is the number of nodes in the graph and C is the number of communities in the graph. F_{ij} represents the strength of node i in community j, i.e., the affiliation of node i to community j. According to this model $p(G|F)$, we can generate the graph G given F. However, overlapping community detection is exactly the opposite task. We have to find the unobserved affiliation matrix F given a graph G [1, 13]. We can optimize F by minimizing some loss function based on the model $p(G|F)$ [1]. On the contrary, we can feed the graph G into a GNN; then we can optimize the GNN to generate a more accurate F [13]. The latter approach performs better in most cases [13]. In this thesis, we follow the GNN-based approach to optimize F.

3.1 Model Overview

We consider an unsupervised approach in this study. Essentially, we design an encoder and decoder-based approach to tackle the problem of overlapping community detection (Figure [3.1]). The encoder takes a graph G as input and generates the community affiliation matrix F as output. The decoder takes F as input and produces a loss. We use the loss generated by the decoder to train the encoder. Indeed, we use a GNN-based encoder [13, 44] and the Bernoulli-Poisson decoder [13]. A shallow GNN with two layers was used in the previous study [13]. However, GNNs can have up to 1000 layers [75]. These deep GNNs are better than shallow GNNs in terms of learning capability [52]. Increasing the number of layers causes some issues such as over-smoothing and the vanishing gradient problems [52]. Consequently, learning completely

fails. In the case of point cloud learning, these issues are resolved by residual connections [53], dynamic edges [52, 61, 76], and dilated aggregation [52, 76] where the considered graph has a fixed neighborhood size. However, in the case of general graphs, the number of neighbors for each node varies widely. Moreover, the information coming from the edge weights of weighted graphs is also crucial. It is still open to incorporating the concept of dilated aggregation, dynamic edge, and the information from edge weights in the case of general graphs (having variable-sized neighborhoods) to achieve deep GCN models. We are the first to introduce these concepts successfully in the case of general irregular graphs to achieve deep GCNs. We term our deep GCN model DynaResGCN. Moreover, we explore different approaches to considering edge-weights of weighted graphs. These approaches combined with DynaResGCN are described in Table 3.1. Finally, we unify the idea of dynamic edges, dilated aggregation, and consideration of edge weights in an encoder-decoder-based framework for overlapping community detection. In the following, we describe our approach to achieving the Deep GCN (DynaResGCN) encoder, the corresponding Bernoulli-Poisson decoder to train the Deep-GCN encoder, and the training algorithm.

3.2 Deep Dynamic Residual GCN encoder

A deep-GCN encoder is a GNN with many layers having the capability of effective learning. However, it is not possible to achieve a deep-GCN by increasing the number of layers in shallow models due to the over-smoothing and vanishing gradient problems [52]. In this study, we introduce a deep-GCN encoder for overlapping community detection in graphs. We incorporate the idea of residual connections, dynamic edges, and dilated aggregation in a single framework which we term DynaResGCN (Figure 3.2). This figure explains the whole architecture of the deep DynaResGCN encoder where message passing of GNN is shown for only a single node (for simplicity). At each layer, information is aggregated from a fixed number of random nodes. Some of these nodes are chosen from the first hop and others are chosen from the second hop neighbors based on the dynamic dilated aggregation algorithm. The descriptions of each part of the architecture are provided in the following text.

3.2.1 Residual Connection

The idea of residual connections was first introduced in CNN for image classification [53]. Later, this idea was transferred into GCN for point cloud learning [52] and has been successfully applied to achieve deep models. In fact, residual connections solve the vanishing gradient problem [53]. In this study, we incorporate residual connections into the GCN encoder for overlapping community detection. If $\hat{A}$ is the normalized adjacency matrix [13], the feature matrix is X_l at layer l and the learnable weight matrix is W_l at layer l, then computation of the

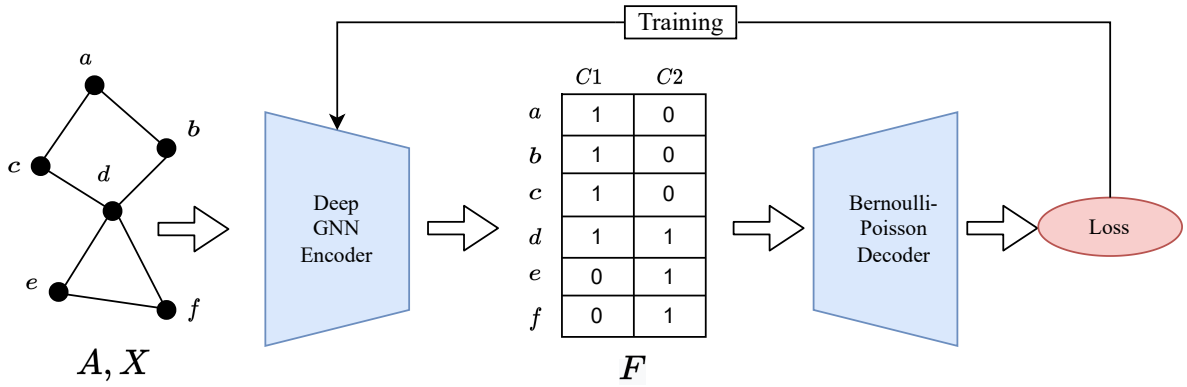


Figure 3.1: The overlapping community detection model based on the deep DynaResGCN encoder and the Bernoulli-Poisson decoder.

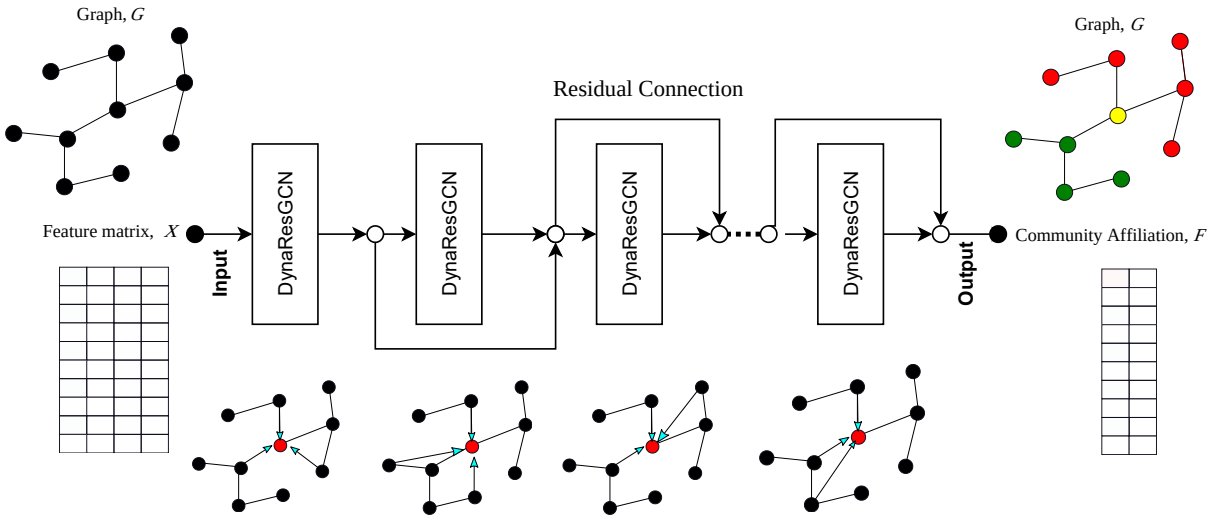


Figure 3.2: The architecture of the deep DynaResGCN encoder.

GCN encoder at layer l is expressed as:

$$X_{l+1} = ReLU(\hat{A}X_l W_l) \tag{3.1}$$

In our study, we incorporated residual connection at layer l, as follows:

$$X_{l+1} = ReLU(\hat{A}X_l W_l) + \hat{A}X_l \tag{3.2}$$
$$= X_{l+1}^{res} + \hat{A}X_l \tag{3.3}$$

The additional term in equation (3.2) is $\hat{A}X_l$. This term connects the input X_l with the output X_{l+1}. Consequently, a direct connection is created in the computation path, which enables smooth flow gradients, and therefore, gradients do not vanish away, which potentially solves the vanishing gradient problem.

3.2.2 Dilated Aggregation

The concept of dilation came from the wavelet processing domain. Specifically, the algorithm named *dilated wavelet convolution* was designed for wavelet processing [77, 78]. For predictive learning, Yu and Koltun [76] proposed dilated convolution for dense prediction tasks. Their aim was to compensate for spatial information loss due to pooling operations. Dilated convolutions were shown to significantly improve the accuracy of dense prediction tasks through aggregating multi-scale contextual information [76]. In fact, dilation increases the receptive field without leading to a loss of resolution [52]. Consequently, dilated aggregation was applied to GNNs for point cloud learning [52]. In the point cloud, a neighborhood is constructed for each point using the k-nearest neighbor (k-NN) approach [52]. Then, dilation is applied through sub-sampling from the fixed neighborhood [52]. For instance, let us assume that $k = 10$. Then the nearest neighborhood of size 10 is constructed from the point cloud for each point. For each point, the nearest neighbors can be ordered based on the distance. To perform dilation, every even-numbered neighbor can be skipped and every odd-numbered neighbor can be considered. Thus, aggregation for information could be done from the first, third, fifth, seventh, and ninth neighbors. As a result, this aggregation is called dilated aggregation. Using dilated aggregation, it is possible to reach distant neighbors without increasing the number of neighbors considered for aggregation.

However, there is no flexibility in the neighborhood size in general graphs. For each node, the neighborhood is defined according to the graph structure. In fact, the size of the neighborhood in general graphs varies widely. Therefore, the fixed-sized k-NN approaches are not applicable to the general graphs. An adaptive approach, that accounts for the variable neighborhood size, is required to be designed. It is also possible to define the neighborhood of the general graphs based on proximity. For instance, we can define first-order proximity based on the first-hop

neighbors of a node. Accordingly, second-order proximity may be defined considering all the first-hop and second-hop neighbors. If we consider the first-order proximity and sub-sample from the first-hop neighbors, there is a possibility of information loss due to losing a number of first-hop neighbors. In the case of higher-order proximity, the neighborhood size can be large and intractable in practical cases. Moreover, a large number of neighbors can boost over-smoothing. Over-smoothing means very similar features for all the nodes in such a way that all the nodes lose distinguishability. To tackle these challenges, we designed novel dilated aggregation algorithms that can be combined with the concepts of dynamic edges and edge weights.

In our design, we have to handle two rival issues. Firstly, dilation in first-order proximity results in information loss. On the contrary, the neighborhood size increases at a quadratic rate when second-order proximity is considered. To achieve a fair balance between these rival issues, we choose the augmented neighborhood size of a node as $2 * m$ where m is the degree of the node. Then, we choose m nodes from the first hop and the remaining m nodes from the second hop. We design augmentation algorithms (Algorithm 1, Algorithm 2) to choose the nodes from the second hop neighbors. Using a weighted augmentation algorithm, we can exploit the information in the weights of the edges to choose neighbors from the second hop. Finally, we introduce dilation in this augmented neighborhood. We have designed different dilation schemes where we can exploit information coming from the edge weights. The simplest dilation scheme is to sub-sample 50% nodes randomly from this augmented neighborhood. After sub-sampling, each node would have m neighbors effectively. Thus, we do not increase the effective neighborhood size while we can aggregate information from the distant nodes.

Let us consider an m-regular graph. In this case, first-order proximity considers a neighborhood of size m. However, the second-order proximity would consider $m^2 + m$ neighbors. In our approach, we consider an augmented neighborhood of size $2 * m$ which is not much larger. To implement dilation, we sample 50% nodes from this augmented neighborhood. Therefore, the effective number of neighbors is $2 * m * 0.5 = m$, which is the same as the number of original first-hop neighbors. Therefore, the possibility of information loss due to dilation is diminished with the advantage of a larger receptive field.

Now, the question arises that how to achieve the augmented neighborhood. It is possible to choose m nodes from the first-hop neighbors. However, the issue is, how to choose the remaining $2 * m - m = m$ nodes from the second-hop neighborhood of size m^2. To resolve this issue, we designed augmentation algorithms, which are explained later. Finally, to implement dilation, we designed different sampling procedures based on various criteria.

3.2.3 Dynamicity of Edge

The dilated aggregation procedures inherently lose some first-hop neighbors. When the same dilated neighborhood is considered at every layer of the GNN, some of the first-hop neighbors

are never explored for each node. This fixed neighborhood at every layer effectively alters the original graph. Moreover, this introduces over-smoothing as the information from the same neighbors is aggregated at each layer. Consequently, learning is not possible. To remedy this issue, we consider different sub-sample of neighbors from the augmented neighborhood at every different layer. Therefore, the neighborhood is different at every layer. In essence, the edge set of the graph becomes dynamic. Interestingly, the concept of dynamic edge and dilated aggregation can be unified. As a result, learning becomes stable and better than static approaches [61, 79, 80]. In the case of point clouds, the nearest neighbor graph is dynamically constructed after every layer [61, 79]. To introduce this concept into the DynaResGCN encoder, the adjacency matrix is constructed differently at each layer based on the augmentation algorithm and dilation scheme. Let us assume A_l is the adjacency matrix at layer l where A_l is different in different layers. Therefore, the computation at layer l defined in (3.2) is updated to the following

$$X_{l+1} = ReLU(\hat{A}_l X_l W_l) + \hat{A}_l X_l \tag{3.4}$$

$$= X^{res}_{l+1} + \hat{A}_l X_l \tag{3.5}$$

Algorithm 1 Graph Augmentation (Random) : Input $G = (V, E)$

for each node u in the graph **do**
  Let S_u be the set of original neighbors of u
  Let S'_u be the set of augmented neighbors of u
  Initialize S'_u with the elements of S_u
  for each node $v \in S_u$ **do**
    Obtain a random node $w \notin S'_u$ from the neighbors of v ($w \in S_v$)
    Assign w in S'_u
  end for
end for
Return $G' = (V, E')$ considering augmented neighborhood

Algorithm 2 Graph Augmentation (Weighted): Input $G = (V, E)$

for each node u in the graph **do**
  Let S_u be the set of original neighbors of u
  Let S'_u be the set of augmented neighbors of u
  Initialize S'_u with the elements of S_u
  for each node $v \in S_u$ **do**
    Obtain the most weighted node $w \notin S'_u$ from the neighbors of v ($w \in S_v$)
    Assign w in S'_u
  end for
end for
Return $G' = (V, E')$ considering augmented neighborhood

3.2.4 Dynamic Dilated Aggregation Schemes

The ideas of residual connection, dilated aggregation, and dynamic edge are unified in a single framework by the equation (3.4). In this section, we define augmentation algorithms and different dilated aggregation techniques. The adjacency matrices $\hat{A}_l$ are generated using these algorithms. We employ two augmentation algorithms: 1) random augmentation (Algorithm 1), and 2) weighted augmentation (Algorithm 2). The random augmentation algorithm is applicable to any graph. However, the weighted augmentation algorithm exploits the information coming from the edge weights of a weighted graph. In Figure 3.3, we show random augmentation and random sampling for a single node. In the following text, we describe different dilated aggregation schemes based on the different sub-sampling/dilation procedures of an augmented neighborhood.

Random Dynamic Dilated Aggregation:

For random dynamic dilated aggregation, at first, we apply Algorithm 1. All the first-hop neighbors are considered in this algorithm. Additionally, for each first-hop neighbor, a new node from the second-hop neighborhood, connected to the first-hop node, is considered. Now, at each layer, we randomly sample 50% of the nodes from the augmented neighbors. The sampling at each layer differs from the sampling at other layers, which allows to explore different neighborhoods. Thus, we employ the concept of dynamic edges, in terms of dynamic neighborhoods, with dilated aggregation. This random dynamic dilated aggregation helps in resolving the over-smoothing problem [52].

Weighted Dynamic Dilated Aggregation:

For weighted dynamic dilated aggregation, we first apply Algorithm 2. All of the first-hop neighbors are considered in this algorithm. Additionally, for each first-hop neighbor, a new node is considered from the second-hop neighborhood connected to the first-hop node. This new node is the most weighted node connected to the first-hop node. The remainder of this process is similar to random dynamic dilated aggregation. In this aggregation scheme, we attempt to exploit the information coming from edge-weights.

Weighted Differential Dilated Aggregation:

In this approach, a weighted augmentation algorithm (Algorithm 2) is used to augment the graph. Then at each layer, importance sampling is performed, in which the first-hop neighbors are given more importance than the second-hop neighbors. However, the amount of priority to be given to the first-hop neighbors over the second-hop neighbors is a hyper-parameter. In our experiment, we consider 80% importance for the first-hop neighbors and 20% importance for the second-hop neighbors. As a result, the first-hop neighbors are selected with a probability of 0.80, and the

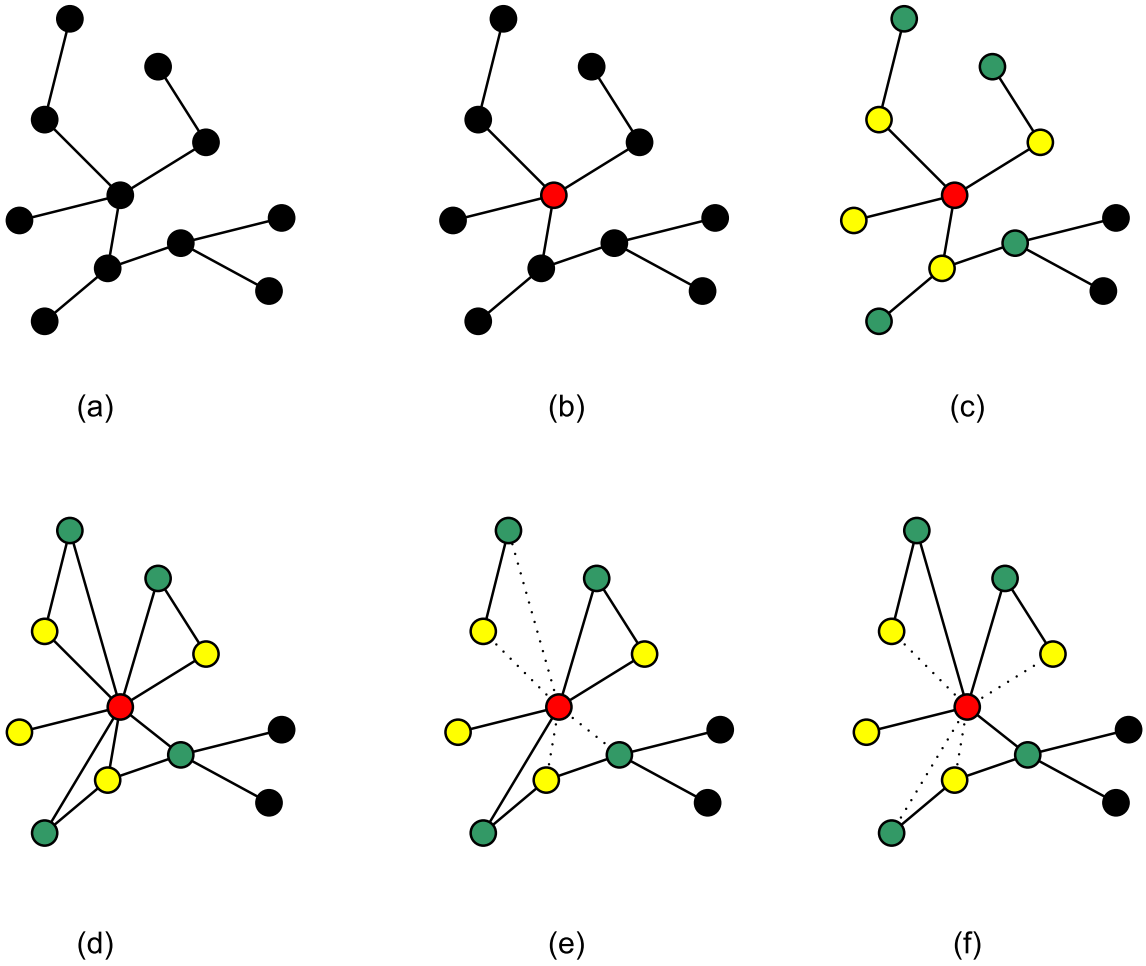


Figure 3.3: (a) A graph (network) (b) in which a node (red) is considered, (c) first-hop neighbors and second-hop neighbors are indicated in yellow and green, respectively, (d) graph augmentation up to second-hop neighbors, (e) dilated neighborhood at one layer (random), and (f) dilated neighborhood at another layer (random)

second-hop neighbors are selected with a probability of 0.20. The intuition behind this step is that if a neighbor is more important to a node, then it should be directly connected to the node. Being a second-hop neighbor automatically implies less importance, irrespective of the weight.

3.3 Bernoulli-Poisson Decoder

In this section, we describe the Bernoulli-Poisson decoder and the process of computing loss function based on the generated affiliation matrix F. Indeed, the final layer of the GCN encoder generates the affiliation matrix, F. To train the GCN, we need to calculate the loss. To generate the loss, we exploit the Bernoulli-Poisson model. In fact, the BigClam model [1] and other works [31, 50] proposed the Bernoulli-Poisson (BP) model. The BP model is a graph generative model. According to this model, for each node u, there is a community affiliation vector F_u with dimension k. This model generates the adjacency matrix, A, for a given community affiliation matrix F, as follows:

$$A_{uv} \sim \text{Bernoulli}(1 - exp(-F_u F_v^T)) \tag{3.6}$$

For a community c and nodes u and v, there is a probability of forming edge between u and v. This probability is related to the affiliation of nodes u and v to community c. That probability is $1 - exp(-F_{uc}.F_{uv})$. Each community has an independent probability of forming edge between every pair of nodes in the graph. In essence, the probability of an edge between u and v, considering every community, is given by $1 - exp(-\sum_c F_{uc}.F_{vc})$ [1]. This BP model can generate nested and hierarchical communities and also can model densely overlapping communities [12, 13]. Yang *et al.* [1] performed maximum likelihood estimation (MLE) to infer communities in the BP model with the coordinate ascent. In contrast, Zhou *et al.* [31] applied a Markov chain Monte Carlo estimation to infer overlapping communities. For our case, we assume F is generated by the GNN encoder. For a given graph with adjacency matrix A and feature matrix X, we generate the community affiliation matrix using the DynaResGCN encoder as follows:

$$F = \text{DynaResGCN}_\theta(A, X) \tag{3.7}$$

Now, according to the BP model, the negative log-likelihood is [13],

$$-\log p(A|F) = -\sum_{(u,v)\in E} \log(1 - exp(1 - F_u F_v^T)) + \sum_{(u,v)\notin E} F_u F_v^T \tag{3.8}$$

Due to the sparsity of many real-life graphs, non-edges are much more common than edges. To overcome this issue, the authors of NOCD [13] proposed balancing the terms of the negative

log-likelihood according to the standard technique for imbalanced data [72].

$$\mathcal{L}(F) = -\mathbb{E}_{(u,v)\sim P_E}[\log(1 - exp(-F_u F_v^T))] + \mathbb{E}_{(u,v)\sim P_N}[F_u F_v^T] \tag{3.9}$$

Here, uniform distributions over edges and non-edges are represented by P_E and P_N respectively. In our case, we follow the same approach as in NOCD [13]. To learn the community affiliation matrix F, the authors of BigClam [1, 73] directly optimized the affiliation matrix F. Unlike these traditional approaches, we optimize the parameters of the GNN to obtain a more accurate affiliation matrix. The equation (3.9) generates the loss. This loss is back-propagated in the GNN encoder to learn the parameters W_l. Next, we describe our training algorithm.

3.4 Training Algorithm

This section describes our end-to-end training algorithm to optimize the deep DynaResGCN encoder for generating better community affiliation matrix. For simplicity, we consider τ as the total number of epochs. In our implementation, we consider early stopping criteria when the learning becomes stable. In step (10) of this algorithm (Algorithm 3), we sometimes vary the

Algorithm 3 Training Algorithm

1: Perform graph augmentation (random, weighted)
2: Build the GNN encoder with a predefined number of layers n
3: For each layer, generate the adjacency matrix $\hat{A}_l$ based on the dilated aggregation scheme
4: **for** each epoch e in τ **do**
5: For stochastic learning, sample a batch edges E.
6: Feed the normalized feature matrix X to the GNN as follows
7: **for** each layer l in total p layers **do**
8: $X_{l+1} = \text{Relu}(\hat{A}_l X_l W_l) + \hat{A}_l X_l$
9: **end for**
10: Get the community affiliation matrix $F = X_p$ from the output of the GNN
11: Calculate loss using equation (3.9)
12: Calculate gradients using this loss
13: Update the network parameters W_l using the gradients
14: **end for**

loss function by adding the conductance to the loss calculated by equation (3.9). Conductance is one of the most important evaluation metrics, as explained later. The lower the conductance value, the better the quality of detected overlapping communities [1]. Therefore, conductance behaves like a loss. Consequently, we can add conductance directly to the loss function.

3.5 Differences with Existing Works

In this section, we discuss differences of our work with the relevant literature. Our method consider the DynaResGCN model as the encoder and the Bernoulli-Poisson model as the decoder. On the decoder perspective, we use the same decoder as the NOCD. However, we work mainly on the encoder part. Our DynaResGCN encoder is a deep GCN model which can handle irregular graphs whereas the state-of-the-art deep GCN model [52] is designed for only regular graphs. Therefore, our model is more suitable for graphs having larger community diameter. We also consider residual connection and edge dynamicity which are not considered in the shallow GCN of NOCD. Other related works such as SLPA, BigClam, DEMON, DANMF, MNMF, etc., do consider any graph convolutional network. Therefore, these works are inherently different from our work. Finally, in Table 3.2, we provide a complete conceptual comparison of our method with the relevant literature and existing works.

Table 3.1: Our studied methods considering different dilated aggregation schemes and loss functions

Method	**Dilation Algorithm**	**Weight**	**Loss Function**
DynaResGCN	random	no	reconstruction error (3.9)
DynaResGCN+cond	random	no	reconstruction error (3.9) + conductance
DynaResGCN+weight	weighted	yes	reconstruction error (3.9)
DynaResGCN+differential weight	weighted differential	yes	reconstruction error (3.9)

Table 3.2: Conceptual comparison of our method to the relevant literature.

Algorithm/ Model	Overlapping community detection	Large community diameter	Residual connection	Edge dynamicity	Dil.Agg. in irregular graphs[1]	Deep GCN model	Reference
DynaResGCN+BP	✔	✔	✔	✔	✔	✔	**Ours**
NOCD	✔	×	×	×	×	×	[13]
DeepGCN	×	N/A	✔	✔	×	✔	[52]
BigClam	✔	✔	×	×	×	×	[1]
SLPA	✔	✔	×	×	×	×	[21]
DEMON	✔	✔	×	×	×	×	[22]
DANMF	✔	✔	×	×	×	×	[2]

(1) Dilated aggregation in the graphs having variable-sized neighborhoods.

Chapter 4

Experimental Study

In this study, we propose different methods based on the different dilated aggregation schemes and different types of graphs (weighted/unweighted). These methods are listed in the Table 3.1. In order to perform evaluation and validation, we have extended the code base of NOCD to implement our ideas. We have two types of datasets: 1) without ground truth, and 2) with ground truth. The research topics dataset ($n \approx 6K$) [54] is a medium-sized dataset without ground truth information. All other datasets have ground-truth information [13, 55]. These datasets with ground truth are of two types: 1) small ($n < 800$) datasets (Facebook datasets [55]), and 2) very large ($n > 20K$ up to $65K$) datasets ([13]). We vary the number of layers from 10 to 90 in steps of 10 for the topics dataset. However, for the small datasets, we vary the number of layers starting from 2 up to 15 in the set $\{2, 3, 5, 7, 10, 15\}$. For the very large datasets, we vary the number of layers up to 50 in the set $\{2, 3, 5, 7, 10, 15, 20, 30, 40, 50\}$. We also consider network widths of 16, 32, 64, or 128. We determine the width for a dataset and use it for all other cases. The threshold to consider a node in a community was also varied in the set $\{0.05, 0.10, 0.125, 0.20, 0.30, 0.35, 0.40, 0.50\}$. For each dataset on each of our considered methods, we find the best model depth and threshold through grid search. Then, for the best hyper-parameter set, we run 50 iterations with different initialization. Finally, we report the mean value of the respective metric over these 50 iterations. In the topics network, we set the number of communities to 100 which is determined based on searching in the set $\{25, 50, 100, 200\}$ and considering the best performing one. For the datasets having ground truth, we set the number of communities as it is in the ground truth (similar to NOCD). We also perform a statistical test to determine the statistical significance in the case of the datasets having ground truth. In the following text, we elaborately describe our baselines, datasets, metrics, statistical testing procedure, experimental setup, results and analyses. Our code is available at https://github.com/buet-gd/Deep-DynaResGCN-community.

4.1 Baselines

In the case of the topics dataset, we use SLPA [21], DEMON [22], MNMF [37], BigClam [1], DANMF [2], and NOCD [13] as the baselines. For NOCD [13], we use their publicly available code. For other baselines, we use the implementations in the CDlib python library with default parameters [81]. In the case of other datasets with ground truth, we compare our methods with the same baselines used in the NOCD [13]. Our method DynaResGCN-G denotes the DynaResGCN method without considering node attributes while DynaResGCN-X denotes the same method considering node attributes. As we consider the exact same baselines on the same datasets as described in NOCD [13], we report the results already published in [13]. These baselines include BigClam [1], EPM [31], SNetOC [50], CESNA [73], SNMF [36], CDE [35], DeepWalk [46] with Non-exhaustive Ovelapping K-means [82](DW/NEO) , and Graph2Gauss [49] with Non-exhaustive Overlapping K-means [82] (G2G/NEO).

4.2 Datasets

All of our datasets are described in Table 4.1. We use a topics graph dataset [54] which is based on various research-topics. We can observe a small portion of the topics network in Figure 4.1. Every topic is a node in the graph. Two topics are connected by an edge if a researcher works on both topics. One of the main motivations of using the topics network is the use of visualization to understand overlapping community structures. Most works in the literature evaluated the results of experiments on community detection by just observing the values of the evaluation metrics. In fact, the similarity/dissimilarity between two different topics is evident from the name of the topics. For instance, if we consider *computer science* and *algorithms*, then it is clearly evident that they should belong to the same community. On the contrary, if we consider *bio-chemistry* and *graph theory*, then it is also obvious that they should not belong to the same community. Thus, we can generate a heatmap of similar topics groups and dissimilar topics groups which would help us understand the results empirically/visually. This heatmap is explained later.

We also consider very large co-authorship networks compiled by authors of NOCD from Microsoft Academic Graph [13]. In these datasets, the communities are assigned roughly based on the respective research areas. In this case, hand-labeled ground truth is not available. Therefore, these datasets are not reliable. However, these datasets may help to understand the scalability of the different methods in the case of very large graphs.

Finally, we consider real-world social network datasets [55]. These are small datasets with reliable ground truth where communities are hand-labeled. Using these datasets we can estimate community recovery with much better reliability.

In essence, we experiment on datasets having different sizes/scales with the number of nodes ranging from 66 up to $65K$. Along with very different sized datasets, we use datasets with

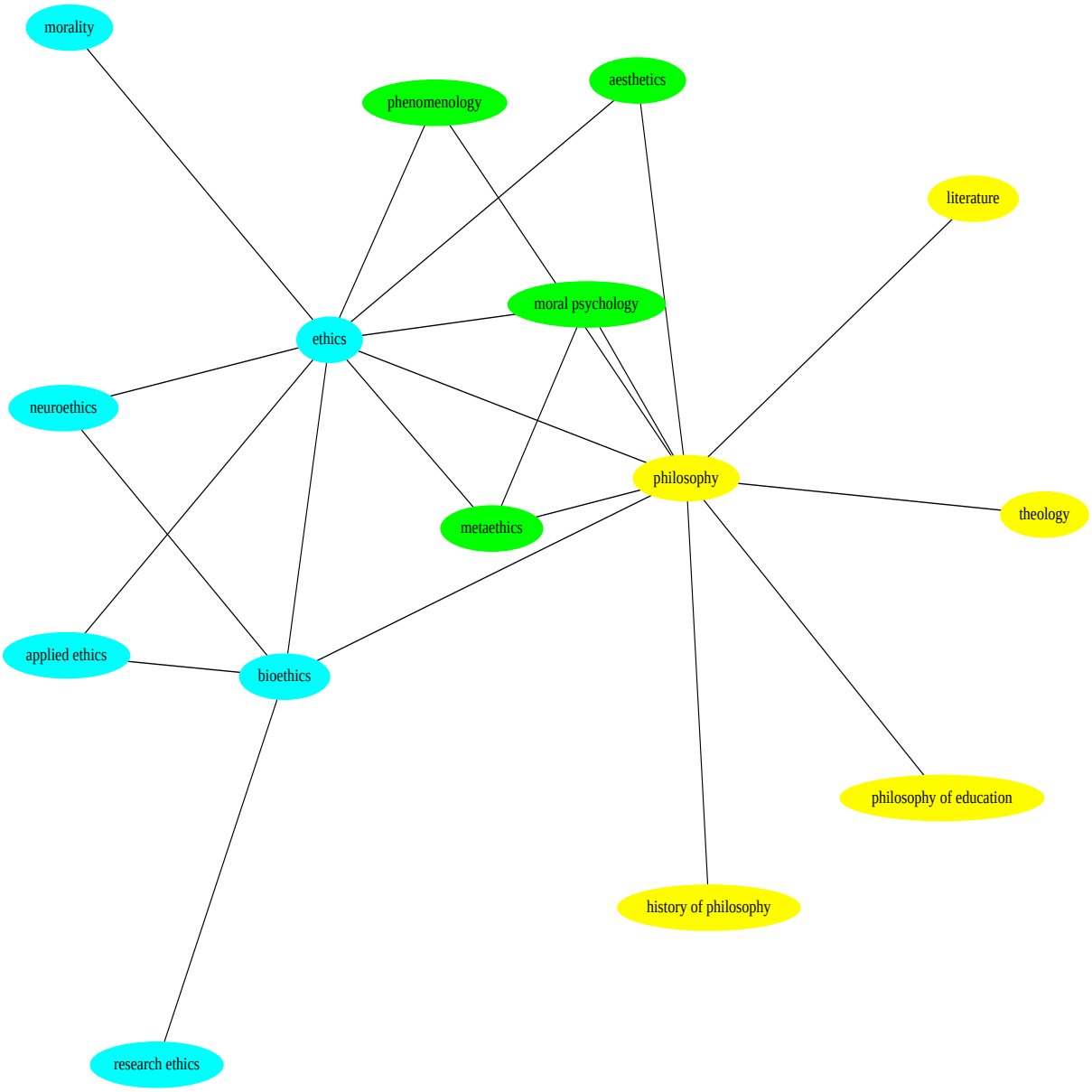


Figure 4.1: Two overlapping communities in a small portion of the topics network.

hand-labeled ground truth, datasets with empirically assigned ground-truth, and datasets without ground truth which ensure the robustness, reliability, and strength of the evaluation procedure.

4.3 Evaluation Metric, Visualization and Statistical Significance Test

To evaluate our methods, we considered several evaluation metrics including conductance, clustering coefficient, density, coverage, normalized mutual information. We also considered a heatmap visualization. Finally, we performed statistical significance tests. These things are described in detail in the following text.

4.3.1 Evaluation Metric

In the case of the topics dataset, we consider different quality measures or fitness functions to evaluate the detected communities. These quality measures include conductance [1], clustering coefficient, density, and coverage. Among these fitness functions, conductance is considered as the main evaluation metric [83]. However, conductance can show very good performance in some corner cases where the actual performance is not good at all. To detect these degenerate cases of conductance, we incorporate other metrics such as the clustering coefficient, density, and coverage. It is impossible that corner cases with very poor actual performance would show good performance in terms of conductance and all other metrics. Among all of the metrics, we consider, conductance is the most important metric, because it captures the basic definition of a community [1]. The higher the density of edges among the internal nodes of a community, and at the same time, the lower the density of edges from the community nodes to the nodes outside of the community, the better community is detected. Conductance captures this definition of a community. However, some degenerate cases can occur in the case of conductance, if the number of communities is very low. In this case, conductance will be very good (very low). However, other metrics will indicate poor performance in such cases. The clustering coefficient captures the percentage of triangles in a community. Thus, this metric measures how strong a community is, without considering the outside nodes of a community. Therefore, if a method performs well in terms of conductance and also comparatively well in terms of all other metrics then this method is a good method. In the case of topics network, we also use heatmaps to understand the overlapping tendency of various communities. In the case of other datasets, we use normalized mutual information (NMI) [57], to measure the similarity between ground-truth communities and predicted communities. In the case of the Facebook datasets, we visualize the largest detected community to understand the quality of the detection. To understand the definitions below, let us assume S is the set of nodes in a single community. $S_1, S_2, \ldots, S_k$ denotes the set of nodes

in different communities. F is the affiliation matrix defined earlier. In the following text, we describe each of the measures and also the visualization processes.

Conductance: We consider the average conductance of the detected communities (weighted by community size); The lower the conductance, the better the performance. In fact, conductance covers the intuitive definition of community [1].

$$\text{Outside}(S) = \sum_{u \in S, v \notin S} A_{uv} \tag{4.1}$$

$$\text{Inside}(S) = \sum_{u \in S, v \in S, v \neq u} A_{uv} \tag{4.2}$$

$$\text{Conductance}(S) = \frac{\text{Outside}(S)}{\text{Inside}(S) + \text{Outside}(S)} \tag{4.3}$$

$$\text{AvgConductance}(S_1, ..., S_k) = \frac{1}{\sum_i |S_i|} \sum_i \text{Conductance}(S_i) \cdot |S_i| \tag{4.4}$$

Coverage: Coverage refers to the percentage of edges explained by at least one community i.e., if (u, v) is an edge and both nodes share at least one community, then this edge is explained by at least one community; the higher the density, the better the performance.

$$\text{Coverage}(S_1, ..., S_k) = \frac{1}{|E|} \sum_{u,v \in E} \mathbf{1}[F_u^T F_v > 0] \tag{4.5}$$

Density: The average density of the detected communities (weighted by community size); the higher the density, the better the performance.

$$\rho(S) = \frac{\text{number of existing edges in } S}{\text{number of possible edges in } S} \tag{4.6}$$

$$\text{AvgDensity}(S_1, ..., S_k) = \frac{1}{\sum_i |S_i|} \sum_i \rho(S_i) \cdot |S_i| \tag{4.7}$$

Clustering Coefficient: The average clustering coefficient of the detected communities (weighted by community size); a higher clustering co-efficient is better.

$$\text{ClustCoef}(S) = \frac{\text{number of existing triangles in } S}{\text{number of possible triangles in } S} \tag{4.8}$$

$$\text{AvgClustCoef}(S_1, ..., S_k) = \frac{1}{\sum_i |S_i|} \sum_i \text{ClustCoef}(S_i) \cdot |S_i| \tag{4.9}$$

Normalized Mutual Information (NMI): It is an information similarity measure between the ground truth communities and the detected communities [57]. We use this measure when we have ground truth information. If we are given two community affiliation matrix $X^{n \times K_x}$ and

$Y^{n\times K_y}$. We have to measure the similarity between these two community structures. To compare community i of X with the community j of Y, we get the column vectors X_i and Y_j. Then we find the following quantities.

$$a = \sum_{m=1}^{n} [X_{im} = 0 \wedge Y_{jm} = 0] \tag{4.10}$$

$$b = \sum_{m=1}^{n} [X_{im} = 0 \wedge Y_{jm} = 1] \tag{4.11}$$

$$c = \sum_{m=1}^{n} [X_{im} = 1 \wedge Y_{jm} = 0] \tag{4.12}$$

$$d = \sum_{m=1}^{n} [X_{im} = 1 \wedge Y_{jm} = 1] \tag{4.13}$$

$$\tag{4.14}$$

If $a + b = n$ and $b = c = 0$, then two vectors X_i and Y_j are in complete agreement. The lack of information in X_i given Y_j is as following:

$$H(X_i|Y_j) = H(X_i, Y_j) - H(Y_j) \tag{4.15}$$

$$= h(a, n) + h(b, n) + h(c, n) + h(d, n) - h(b + d, n) - h(a + c, n) \tag{4.16}$$

where $h(w, n) = -w \log_2 \frac{w}{n}$. However, this measure has a problem that if two vectors X_i and Y_j become near complement of each other, then mutual information will be high. It is not expected. We want that mutual information should be zero in this case. Thus it is refined as follows:

$$H^*(X_i|Y_j) = \begin{cases} H(X_i|Y_j) & \text{if } h(a, n) + h(d, n) \geq h(b, n) + h(c, n) \\ h(c + d, n) + h(a + b, n) & \text{otherwise.} \end{cases} \tag{4.17}$$

This allows us to compare two vectors. Now we compare X_i with the entire Y matrix as follows,

$$H(X_i|Y) = \min_{j\in\{1,\ldots,K_y\}} H^*(X_i|Y_j). \tag{4.18}$$

now we compare two matrices X and Y as

$$H(X|Y) = \sum_{i\in\{1,\ldots,K_x\}} H(X_i|Y) \tag{4.19}$$

We define $H(Y|X)$ as we have defined $H(X|Y)$ where the roles of X and Y is reversed. Now

we define whole entropy of X as,

$$H(X) = \sum_{i=1}^{K_x} H(X_i) \tag{4.20}$$

$$= \sum_{i=1}^{K_x} \left(h\left(\sum_{m=1}^{n} [X_{im} = 1], n \right) + h\left(\sum_{m=1}^{n} [X_{im} = 0], n \right) \right). \tag{4.21}$$

We need the following identities based on the Figure 4.2.

$$H(X) = I(X:Y) + H(X|Y) \tag{4.22}$$

$$H(Y) = I(X:Y) + H(Y|X) \tag{4.23}$$

$$H(X,Y) = H(X) + H(Y|X) \tag{4.24}$$

$$H(X,Y) = H(Y) + H(X|Y) \tag{4.25}$$

$$H(X,Y) = I(X:Y) + H(X|Y) + H(Y|X) \tag{4.26}$$

Mutual information is defined as

$$I(X:Y) = \frac{1}{2}[H(X) - H(X|Y) + H(Y) - H(Y|X)] \tag{4.27}$$

Now our normalized mutual information

$$NMI_{max} = \frac{I(X:Y)}{\max(H(X), H(Y))} \tag{4.28}$$

This normalized mutual information is more accurate than many previous methods like NMI_{LFK}, $RandIndex$, $OmegaIndex$, etc [84]. Most previous works considered NMI as a comparison metric whenever the ground truth available [13]. This NMI metric reduces unintuitive behaviour of previous metrics.

4.3.2 Heatmap Visualization

In the case of the topics dataset, we know the name of the topics which are similar and the name of the topics which are dissimilar. To generate each heatmap we consider a total of nine topics. Among these nine topics, every three topics are similar. Thus we have three groups of similar topics. However, two topics from two different groups should be dissimilar. Then, for each topic, we find the community with the strongest affiliation (as described earlier, we generate community strength for each community for each node). Now, we know the strongest community for each topic. Therefore, we have nine communities corresponding to the nine topics. Then, we find the number of overlapped nodes for each pair of the communities which can be presented as an overlapping matrix where communities are indexed in the same order both in rows and columns.

Table 4.1: Our considered datasets

Dataset	**Nodes**	**Edges**	**MCD**[1]	**Connectivity**[2]	**Ground truth**	**Comment**
Topic	5947	26695	N/A	✔	Not available	Research topics network [54]
Facebook 348	227	6384	4	×	Hand-labeled	Social network [55]
Facebook 414	159	3386	2	×	Hand-labeled	Social network [55]
Facebook 686	170	3312	5	×	Hand-labeled	Social network [55]
Facebook 698	66	540	2	×	Hand-labeled	Social network [55]
Facebook 1684	792	28048	5	×	Hand-labeled	Social network [55]
Facebook 1912	755	60050	3	×	Hand-labeled	Social network [55]
Computer Science	21957	193500	INF	✔	Empirical[3]	Co-authorship network [13]
Engineering	14297	98610	INF	✔	Empirical[3]	Co-authorship network [13]
Medicine	63282	1620628	INF	✔	Empirical[3]	Co-authorship network [13]

(1) MCD stands for the maximum community diameter. N/A refers to that ground truth is unavailable. INF refers that every community-induced sub-graph in the corresponding dataset is a disconnected graph. Therefore, MCD is undefined and theoretically infinite.
(2) Connectivity says whether the whole graph is connected.
(3) Community assignment is not hand-labeled by a human agent.

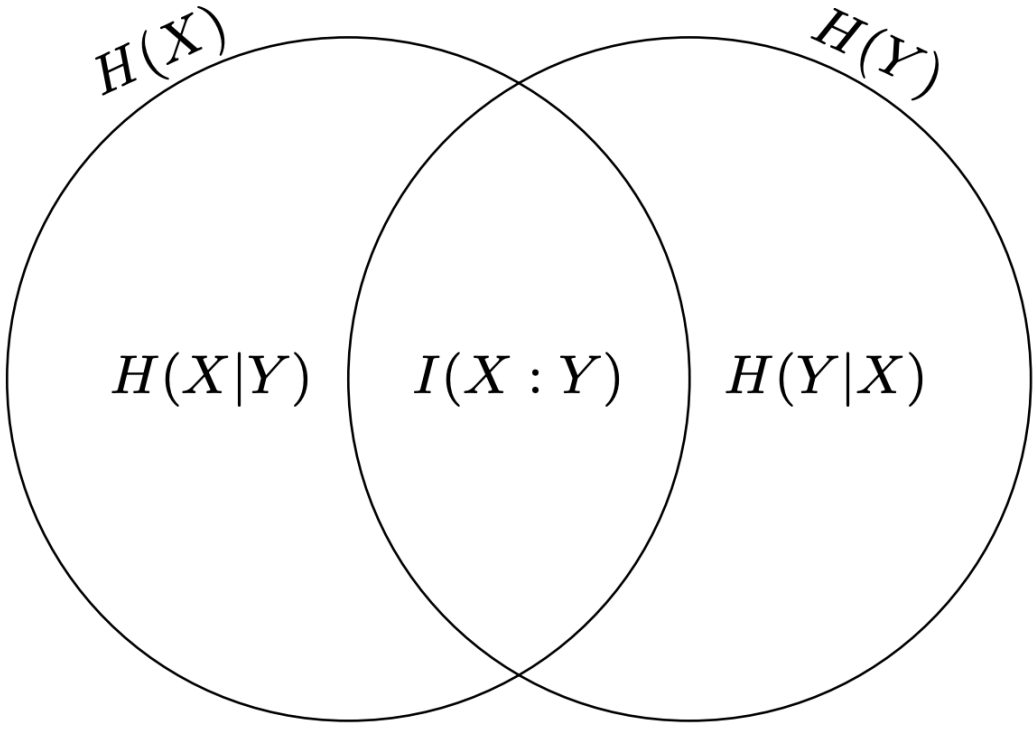


Figure 4.2: Mutual information and variation of information

Finally, we normalize this overlapping matrix. This normalized overlapping matrix is shown as a heatmap. Each cell of the heatmap shows the overlapping strength between the corresponding communities. The higher the overlapping, the darker the cell.

4.3.3 Statistical Significance Test

We use NMI in the case of the datasets having ground truth. We use a statistical t-test to measure the significance of the difference in mean NMI between NOCD and DynaDesGCN. We perform a standard t-test with 95% confidence (significance $\alpha = 0.05$) to test whether the difference between two means from two different distributions is statistically significant [85, 86]. Let $\bar{x_1}$ is the mean over $n_1 = 50$ (as we run 50 independent initializations for each method) independent samples from one method where s_1 is the standard error and $\bar{x_2}$ is the mean over $n_2 = 50$ different samples from another method where s_2 is the standard error. Then, our null hypothesis (H_0) is that the difference in the mean is zero ($\mu_1 = \mu_2$) or statistically not significant. The alternative hypothesis is that the difference in the mean is statistically different $\mu_1 \neq \mu_2$. Now, we calculate the standard t-statistic as follows:

$$t = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{4.29}$$

For a t-test with a significance level of $\alpha = 0.05$, we calculate upper sided critical value from the t-distribution with degrees of freedom 49 ($\min(n_1 - 1, n_2 - 1)$). Let us assume t^* is the $\alpha/2$ upper critical value ($(1 - \alpha/2) * 100 = 97.5$th percentile). If the absolute t-statistic is greater than the t^*, then we reject the null hypothesis and state that the difference of means is statistically significant.

4.4 Experimental Setup

We perform our experimentation on GTX GPU in the Linux platform. Some of the experiments are performed in the Cyverse cloud computing platform [87]. We have implemented our codes in the PyTorch library with Python version 8. We use weight decay rate as 0.01, learning rate 0.001, batch-normalization, and stochastic loss [13] based on the sampled edges of the original graph during training. We also use the early stopping criterion for training with a maximum of 500 epochs.

4.5 Results

A previous study [13] demonstrated the effectiveness of graph neural networks for detecting overlapping communities. They showed that GNNs perform better than free variable optimization or other direct methods such as BigClam [1, 13], EPM [31], SNetOC, [50] etc. In this study, we aim to develop a better GNN-based method to detect overlapping communities with higher detection quality. Specifically, we design a deep GCN encoder considering dynamic dilated aggregation for the overlapping community detection model [13]. Eventually, we achieve significantly better results and detection quality compared to other related studies. For evaluation, we use three different types of datasets. In the following text, we describe our results.

4.5.1 Results on Dataset without Ground Truth

Table 4.2 compares the performance of our various methods with the existing various methods on topics dataset [54] which has no ground truth. In practice, there are no sufficiently large datasets with reliable overlapping ground truth information. Overlapping ground truth information is also more challenging because of subjectivity. It is not impossible to have different ground truths for the same dataset if the dataset is labeled based on two different perspectives. Moreover, the amount of overlap between two communities is also subjective. Indeed, large graphs with hand-labeled ground truth are not openly available. Therefore, though some authors [13] claim they have overlapping ground truth datasets with large graphs, these are not reliable. Moreover, these datasets are also not hand-labeled. As a result, it is necessary to evaluate different methods on unsupervised (having no ground truth) datasets with reliable evaluation metrics. In fact, it is also good to assess with some auxiliary metrics/fitness functions to detect the corner cases. Therefore, in this study, we use conductance as the main evaluation metric [1], as well as three auxiliary metrics to determine whether the main metric (conductance) is actually performing well. Finally, we attempt to visualize overlapping tendencies using a heatmap.

We compare our methods, *DynaResGCN*, *DynaResGCN+cond*, *DynaResGCN+weight*, and *DyanResGCN+differential weight* with NOCD [13], SLPA [21], BigClam [1], DANMF [2], DEMON [22], and MNMF [37] on the topics dataset. The results are presented in Table 4.2. In terms of conductance, DynaResGCN performs the best (the lower the conductance, the better the performance). Bigclam has the closest performance to DynaResGCN in terms of conductance. However, in terms of clustering co-efficient and density, BigClam performs the worst. In terms of coverage, the performance of BigClam is also not good. Therefore, the performance of BigClam, in this case, is a degenerate case. Among all of the metrics, we consider, conductance is the most important metric because it captures the basic definition of a community [1]. The higher the density of edges among the internal nodes of a community, and at the same time, the lower the density of edges from the community nodes to the nodes outside of the community, the better community is detected. Conductance captures this definition of a community. However, some

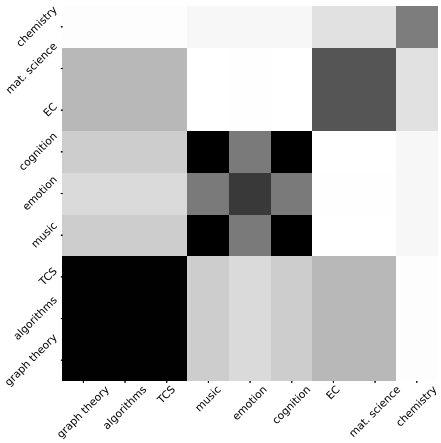
degenerate cases can occur in the case of conductance, if the number of communities is very low. In this case, conductance will be very good (very low). However, other metrics will indicate poor performance in such cases. This situation happens for BigClam, which detects only six communities (automatically). The clustering coefficient captures the percentage of triangles in a community. Thus, this metric measures how strong a community is, without considering the outside nodes of a community. Therefore, if a method performs well in terms of conductance and also comparatively well in terms of all other metrics then this method is a good method. Our best method DynaResGCN performs the best in terms of conductance and also comparatively well in terms of the other metrics.

Instead of relying on just numbers, we attempt to understand overlapping community detection through a novel heatmap visualization approach. We choose three groups of topics to create heatmaps based on the topic network dataset. The first group consists of *graph theory*, *algorithms*, and *theoretical computer science*. The second group consists of *music*, *emotion*, and *cognition*. The third group consists of *electro-chemistry*, *material science*, and *chemistry*. In the heatmap in Figure 4.3, the topics are arranged in order of the first group, second group, and third group.

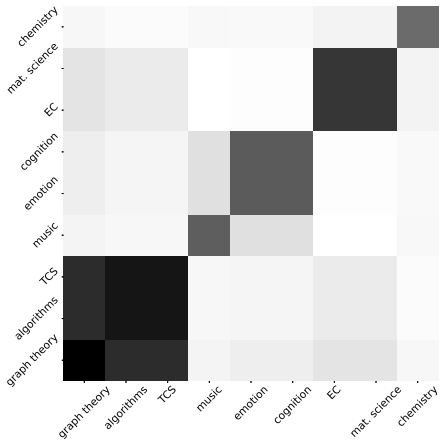
The heatmaps show the effectiveness of each method for detecting overlapping regions. It is evident from the heatmaps that the GNN-based models are very effective for detecting overlapping communities. It is also evident from the heatmaps that the DynaResGCN-based models better detect overlapping communities than the shallow models. There is an interesting observation related to the BigClam model. Based on conductance, BigClam performed very well. However, in terms of the other metrics, BigClam performed poorly. The heatmaps clearly reveal that the performance of BigClam based on conductance is a corner case or degenerate case, where conductance does not reflect the true phenomenon. Therefore, based on the heatmaps, we confidently conclude that if a method performs well in terms of conductance and also in all other metrics, *only* then we can refer that the method is a good/better method. In fact, we are the first to visualize overlapping communities using heatmaps to better understand the detection of overlapping communities.

Finally, if we consider the performance with weighted dilated aggregations, then the performance is almost similar to DynaResGCN. However, in terms of the heatmap, the weighted method seems to have a better heatmap. Therefore, it is reasonable to consider weighted dilated aggregations. It also remains a future work to consider weights in the DynaResGCN model in a more meaningful way.

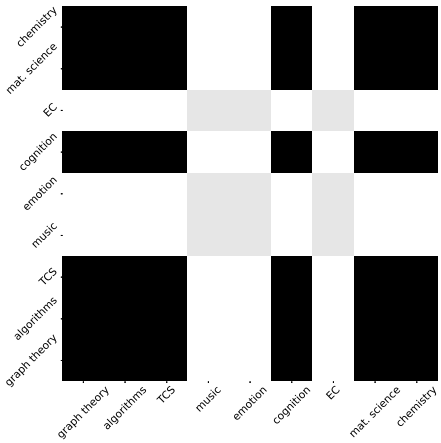
We also used another visualization technique to assess the quality of clustering. In this approach, we marked the largest cluster identified by each method. The largest cluster should not contain the whole network and, also, should not contain a very small portion of the entire network (graph). The weighted model, unweighted model, shallow model, and SLPA show natural/intuitive behavior in terms of the largest cluster identification. However, BigClam includes the whole graph in the largest cluster. NMNF, DEMON, and DANMF include very small portions of the



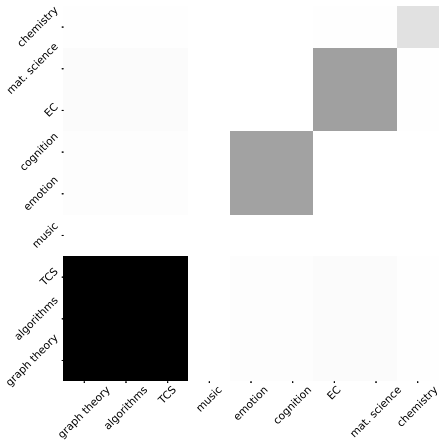
(a) DynaResGCN (**ours**)



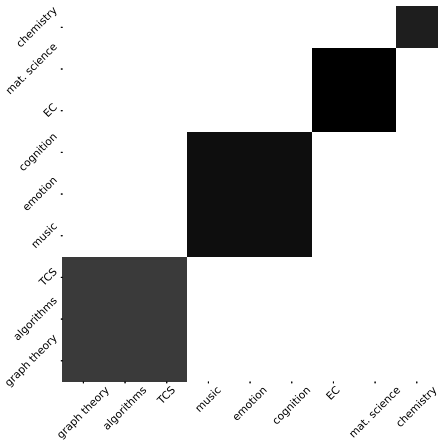
(b) NOCD



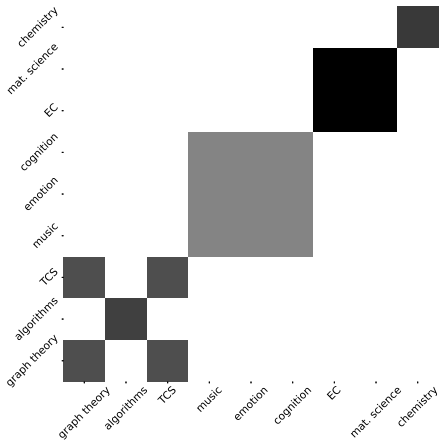
(c) BigClam



(d) SLPA



(e) NMNF



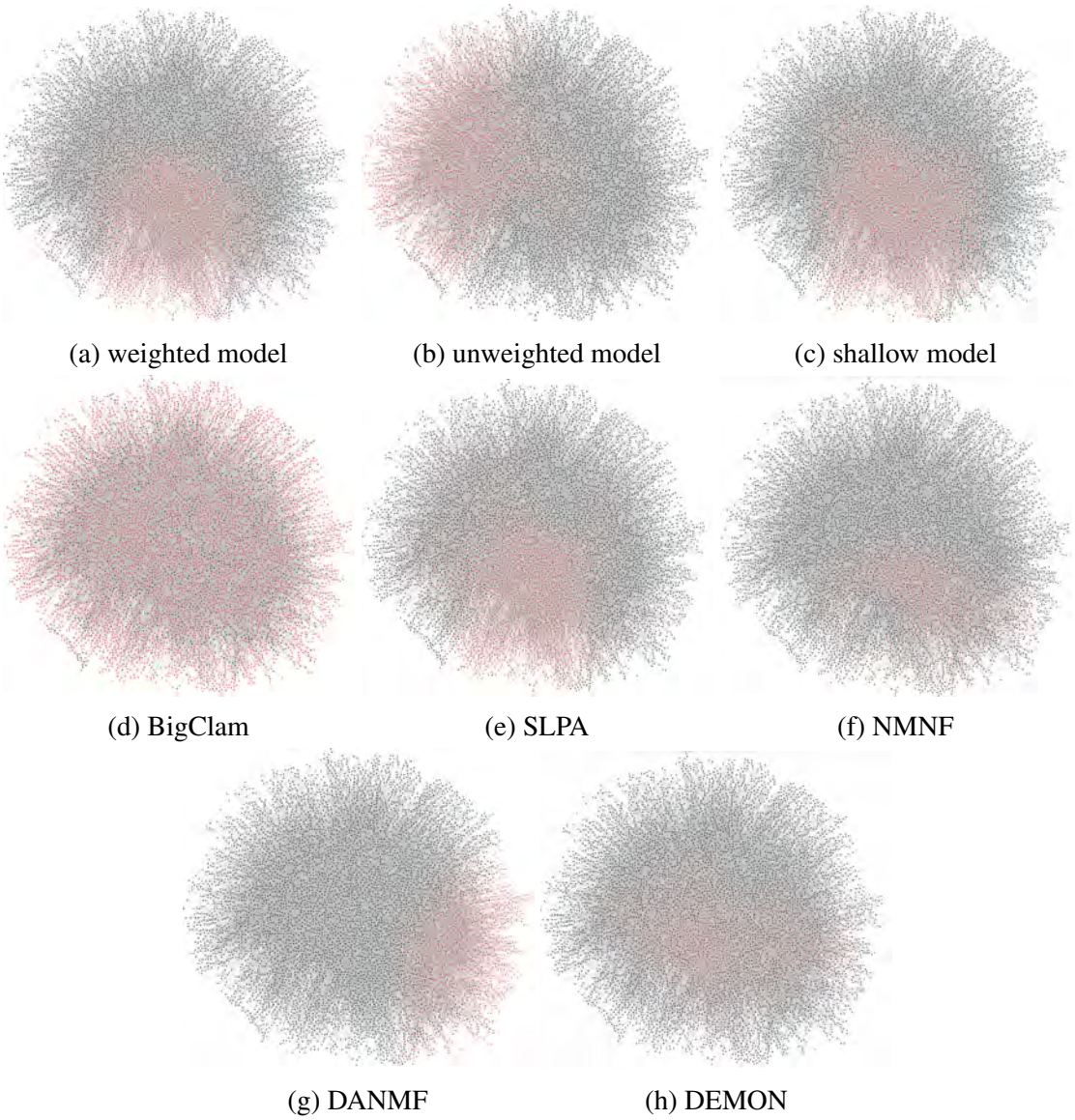
(f) DANMF

Figure 4.3: Overlapping tendency of selected topics from the topics dataset. The darker the cell, the higher the overlap.

whole graph as the largest cluster. Therefore, it is evident that the GNN-based models perform better in community detection.

Table 4.2: **Results from Topic dataset**

Method	Conductance (↓)	clustering coefficient (↑)	Density (↑)	Coverage (↑)
DynaResGCN+cond	0.2144	0.00021	0.00759	0.9984
DynaResGCN	**0.1807**	0.00037	0.00781	0.9986
DynaResGCN+weight	0.1957	0.00023	0.00767	0.9989
DynaResGCN+differential weight	0.2126	0.00019	0.00652	0.9985
NOCD [13]	0.3049	0.00016	0.01135	**0.9999**
SLPA [21]	0.3058	**0.00217**	**0.04867**	0.8273
BigClam [1]	0.2129	0.00000	0.00175	0.8698
DANMF [2]	0.2753	0.00008	0.00869	0.7250
DEMON [22]	0.4873	0.00151	0.04199	0.8965
MNMF [37]	0.3965	0.00009	0.00907	0.6179



(a) weighted model (b) unweighted model (c) shallow model

(d) BigClam (e) SLPA (f) NMNF

(g) DANMF (h) DEMON

Figure 4.4: The largest cluster identified from each method in TOPIC dataset

4.5.2 Results on Datasets with Ground Truth

We compare our best method DynaResGCN model on a set of hand-labeled graphs/networks and on a set of very large graphs with empirically assigned ground truth. In both of the cases, we compare the DynaResGCN and NOCD with attributes of the nodes and without attributes of the nodes in terms of NMI. To evaluate in terms of NMI, we require ground truth. The methods DynaResGCN-G and NOCD-G are compared without considering the attributes of the nodes in Table 4.3 and the details on DynaResGCN-G are described in Table 4.4 . On the other hand, the methods DynaResGCN-X and NOCD-X are compared considering node attributes in Table 4.5 and details on DynaResGCN-X are described in Table 4.9. We explain the results in the following text:

Dataset with Hand-labeled Ground Truth

These are small graphs from Facebook [55] with reliable (hand-labeled) ground truth. From the Table 4.3, and Table 4.5, it is clearly evident that our method DynaResGCN significantly outperforms the nearest best method NOCD in all of the Facebook networks irrespective of the node attributes. In fact, only in the case of the dataset *Facebook 414* NOCD-X has comparative performance with DynaResGCN-X. In these datasets with ground truth, DynaResGCN outperforms all the other methods by a large margin.

One of the key implications of our method is the robustness to node features. In fact, in some cases, DynaResGCN-G outperforms DynaResGCN-X. For instance, in the datasets including *Facebook 348, Facebook 698, Facebook 1684, and Facebook 1912*, DynaResGCN-G is surprisingly robust outperforming all the methods even with node attributes. This implies that our method can exploit structural information lying in the graph very well. From Table 4.4 and Table 4.9, it is prominent that different datasets have different model depths and thresholds which in turn proves that model depth and thresholds are subjective to the datasets.

Achieving the state-of-the-art performance from deeper models implies that our method can train deep GCNs successfully in the general irregular graphs. This proves that DynaResGCN can resolve the over-smoothing problem and vanishing gradient problem. It also proves that we are successful to incorporate the concepts of residual connection, dilated aggregation, and dynamic edges in our DynaResGCN model effectively.

We also attempt to understand the results from the Facebook dataset by visualizing the largest cluster. Interestingly, the GNN-based methods can clearly identify the largest cluster. Among the GNN-based methods, the DynaResGCN-X model identified the largest cluster in a clearer way (Figure 4.5a). The NOCD-X (GNN-based) model can identify the largest cluster in a fairly good manner. However, in Figure 4.5b, we mark some part that should be in the largest cluster but is excluded from the largest cluster by NOCD. Other methods like SLPA and DEMON clearly fail to identify the largest cluster as in Figure 4.5c, 4.5d. It is evident that the DynaResGCN based

Table 4.3: Results for community recovery in terms of overlapping NMI without node attributes using 50 different initializations.

Dataset	BigCLAM	CESNA	EPM	SNetOC	CDE	SNMF	DW/NEO	G2G/NEO	NOCD-G	DynaResGCN-G	Significant[1]
Facebook 348	26.0	29.4	6.5	24.0	24.8	13.5	31.2	17.2	34.7	**39.8**	✔
Facebook 414	48.3	50.3	17.5	52.0	28.7	32.5	40.9	32.3	56.3	**58.1**	✔
Facebook 686	13.8	13.3	3.1	10.6	13.5	11.6	11.8	5.6	20.6	**25.4**	✔
Facebook 698	45.6	39.4	9.2	44.9	31.6	28.0	40.1	2.6	49.3	**51.0**	✔
Facebook 1684	32.7	28.0	6.8	26.1	28.8	13.0	37.2	9.9	34.7	**44.3**	✔
Facebook 1912	21.4	21.2	9.8	21.4	15.5	23.4	20.8	16.0	36.8	**40.1**	✔
Computer Science	0.0	33.8	DNF	DNF	DNF	9.4	3.2	31.2	34.2	**37.4**	✔
Engineering	7.9	24.3	DNF	DNF	DNF	10.1	4.7	33.4	18.4	**37.3**	✔
Medicine	0.0	14.4	DNF	DNF	DNF	4.9	5.5	28.8	27.4	**37.3**	✔

(1) The significance of the mean difference of NMI between NOCD-G and DynaResGCN-G is determined with 95% confidence based on the standard t-test.

Table 4.4: DynaResGCN-G experimental details and results

Dataset	Mean NMI	Std error (NMI)	Layers	Threshold
Facebook 348	39.80	± 2.2	7	0.40
Facebook 414	58.02	± 3.2	2	0.50
Facebook 686	25.40	± 1.9	7	0.35
Facebook 698	51.00	± 3.3	5	0.50
Facebook 1684	44.30	± 2.6	3	0.50
Facebook 1912	40.10	± 1.7	3	0.50
Computer Science	37.4	± 1.7	15	0.50
Engineering	37.3	± 1.2	15	0.35
Medicine	37.3	± 1.1	15	0.35

Table 4.5: Results for community recovery in terms of overlapping NMI considering node attributes using 50 different initializations.

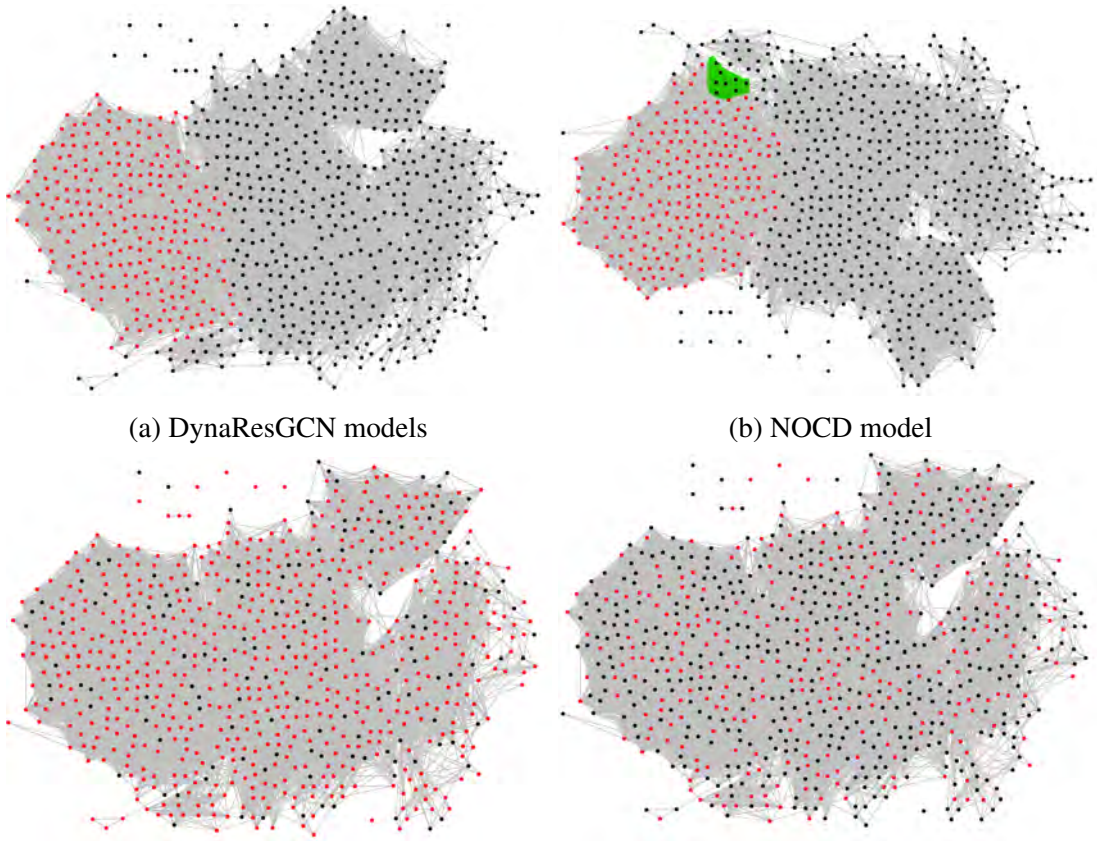
Dataset	BigCLAM	CESNA	EPM	SNetOC	CDE	SNMF	DW/NEO	G2G/NEO	NOCD-X	DynaResGCN-X	Significant[1]
Facebook 348	26.0	29.4	6.5	24.0	24.8	13.5	31.2	17.2	36.4	**39.8**	✔
Facebook 414	48.3	50.3	17.5	52.0	28.7	32.5	40.9	32.3	**59.8**	**59.0**	×
Facebook 686	13.8	13.3	3.1	10.6	13.5	11.6	11.8	5.6	21.0	**27.3**	✔
Facebook 698	45.6	39.4	9.2	44.9	31.6	28.0	40.1	2.6	41.7	**50.1**	✔
Facebook 1684	32.7	28.0	6.8	26.1	28.8	13.0	37.2	9.9	26.1	**41.0**	✔
Facebook 1912	21.4	21.2	9.8	21.4	15.5	23.4	20.8	16.0	35.6	**39.6**	✔
Computer Science	0.0	33.8	DNF	DNF	DNF	9.4	3.2	31.2	**50.2**	46.0	✔
Engineering	7.9	24.3	DNF	DNF	DNF	10.1	4.7	33.4	**39.1**	**39.0**	×
Medicine	0.0	14.4	DNF	DNF	DNF	4.9	5.5	28.8	37.8	**40.0**	✔

(1) The significance of the mean difference of NMI between NOCD-X and DynaResGCN-X is determined with 95% confidence based on the standard t-test.

Table 4.6: DynaResGCN-X experimental details and results

Dataset	Mean NMI	Std error (NMI)	Layers	Threshold
Facebook 348	39.8	± 2.2	7	0.35
Facebook 414	59.0	± 3.1	3	0.50
Facebook 686	27.3	± 2.0	5	0.125
Facebook 698	50.1	± 2.3	7	0.50
Facebook 1684	41.0	± 2.1	7	0.50
Facebook 1912	39.6	± 1.7	3	0.50
Computer Science	46.0	± 2.2	2	0.50
Engineering	39.0	± 3.3	40	0.35
Medicine	40.0	± 2.1	2	0.35

method is clearly the best method to identify overlapping communities in graphs/networks.



(a) DynaResGCN models (b) NOCD model

(c) DEMON (d) SLPA

Figure 4.5: The largest cluster identified by each method in *Facebook 1912* dataset

Datasets with Empirical Ground Truth

These datasets include *Computer Science*, *Engineering*, and *Medicine*. These are very large datasets having nodes up to $65K$. Reliable ground truth information is not available for these datasets. As these are co-authorship networks, ground truth is roughly assigned based on the research area. In this case, DynaResGCN outperforms NOCD by a large margin when node attributes are ignored. When node attributes are considered, NOCD can outperform DynaResGCN only in a single case. In most other cases, DynaResGCN significantly outperforms NOCD. Moreover, it also proves that DynaResGCN is much more robust to node attributes. In fact, in some cases, DynaResGCN can reach the best performance even without considering node attributes.

4.5.3 Execution Time

The execution times for NOCD and DynaResGCN are reported in Table 4.7. In this table, pre-processing time refers to the time for graph augmentation before training. In the case of DynaResGCN method, the execution time is reported for the optimal number of layers for each dataset as it is in the Table 4.4. For the smaller datasets from Facebook, the execution time for DynaResGCN is less 15 seconds even with deep model upto seven layers. In the case of very large datasets upto 65 thousands nodes, the execution time of DynaResGCN is less than 15 minutes with deep model upto 15 layers. In fact, the execution time of DynaResGCN is less than 2 minutes even with 15 layers except for the largest dataset (*Medicine*).

4.5.4 Theoretical Complexity

In this section, we determine the theoretical computational complexity of our DynaResGCN encoder. For simplicity, we ignore graph augmentation before pre-training. We would like to find the computational complexity of each iteration of the forwarding pass of the algorithm to generate the community affiliation matrix from the given graph and feature matrix. We consider M as the number of edges, D the dimension of the input feature space for each node, C is the number of communities, W is the network width in every layer except the output layer, L is the total number of hidden layers. As we sample 50% of the nodes at each layer from the augmented neighborhood, the effective node degree distribution remains the same. Thus the total number of effective edges for information aggregation in each layer is the same as M. Now, we calculate the computational complexity over each effective edge for information aggregation. The computational complexity from the input layer to the first hidden layer is $\mathcal{O}(DW)$. For each internal hidden layer, it is $\mathcal{O}(W^2)$, and for the final hidden layer to the output layer it is $\mathcal{O}(WC)$. Thus, the overall complexity for an edge is $\mathcal{O}(DW + (L-1)W^2 + WC)$ as the computation of internal layers is repeated $(L-1)$ times. Thus overall time complexity considering all the edges

is,

$$\mathcal{O}(M(DW + W^2(L-1) + WC)) \tag{4.30}$$

As we only augment the number of edges by two times for each node, thus memory complexity would be,

$$\mathcal{O}(M). \tag{4.31}$$

Our overall computational complexity depends on two things: 1) the number of hidden layers (model depth) and 2) the number of communities. However, the effect of model depth is more dominant than the effect of the number of communities. In real life, the number of communities is unknown in advance and hence we consider a constant number of communities for every case. In that case, we can ignore the effect of the number of communities.

4.6 Analyses

In the following text, we discuss our insights learned from the experimentation and results. We analyze the effects different aspects of our model in detail. These includes model depth, residual connection, dynamic dilated aggregation, edge weights, threshold sensitivity, network comlexity, etc. We also attempt to find out our limitations and subsequent future directions.

4.6.1 Model Depth

From the experiments on the different datasets, we find that the best model depth is subjective to the dataset. Sometimes we find that the smaller datasets have the best results with a smaller number of layers and comparatively larger dataset such as the topics dataset achieves the best results with a large number of layers. The reason behind this is that the smaller datasets have smaller community sizes while the larger datasets have larger community sizes. It is evident that the deeper the GCN model, the more distant neighbors are aggregated. Therefore, the deep model can aggregate information from distant neighbors, which is required to discover a larger community. Accordingly, when the community size is smaller, we should explore closer neighbors which require less deep models. It is clearly evident in Figure 4.6 that the best model depth is highly correlated with the maximum community diameter (MCD). Even in some cases model depth is exactly the same as MCD. The key insight here is that we should not define a fixed depth for the model beforehand, rather we should vary the depth of the model depending on the nature of the dataset. This implies that the GCN model depth is subjective to the dataset. However, training deeper models is not trivial. We have to overcome several key issues in the course of training our Deep DynaResGCN model. One of the key challenges is

the vanishing/exploding gradient problem which is resolved by residual connection. Another challenge includes the over-smoothing problem. We have resolved this over-smoothing problem using randomized dynamic aggregation in the deep DynaResGCN model. In the future, It would be interesting to explore the relationship between model depth and community detection from a theoretical perspective.

4.6.2 Dynamic Dilated Aggregation

However, then one question arises why should we need the small depth DynaResGCN model for the smaller datasets? In fact, the DynaResGCN model achieves superior performance than the NOCD model in the case of smaller datasets also. Here comes the power of the dynamic dilated aggregation scheme in the DynaResGCN model. Because of the dilated aggregation in the DynaResGCN model, it can aggregate information from the second hop neighbors in one layer of the GCN without increasing the effective number of neighbors (explained earlier). It is the dynamicity coming from the random sampling of the augmented neighbors, which allows exploring a number of new neighbors at every different layer because we perform a new sub-sampling of the neighbors at every different layer. Therefore, the model gets some new information at every layer. In fact, dilated aggregation mechanism also helps to alleviate the over-smoothing problem [52]. Moreover, weighted dilated aggregation (in weighted graphs) allows for the exploitation of the information lying in the edge weights. Hence, dynamic dilated aggregation is one of the key ideas to achieving better results in the DynaResGCN model.

One of the most important effects of dynamicity of aggregation is the introduction of variance due to randomness. This variance is one of the keys to resolving the over-smoothing problem. On the other hand, this variance of node features introduces the distinguishability of the nodes which may help to identify different clusters. Another source of randomness may be the introduction of some random noises while aggregating. Let us assume ζ is a random noise vector with dimension d chosen from a zero-mean normal distribution as follows:

$$\zeta \sim \mathcal{N}(\mathbf{0}, \mathbf{I}\sigma^2)$$

Here we consider uniform variance σ^2 over all the dimensions. For n nodes of the graph, we may consider a matrix of random noises as $\Sigma^{n\times d}$ which would be added to the features matrix before each aggregation step as follows:

$$X = X + \Sigma$$

This approach will ensure a provable variance of σ^2 to each value. In the future, we have plans to explore this idea to reduce over-smoothing in deep GCNs. Interestingly, if we follow the same distribution to sample noise vectors, the overall noise would be zero in expectation with

Table 4.7: Comparison of execution time of NOCD and DynaResGCN without node attributes. Time of DynaResGCN is reported for the best number of layers as in table 4.4. All the times are reported in seconds.

Dataset	NOCD	DynaResGCN	
		training	pre-processing + training
Facebook 348	6	12	12
Facebook 414	5	5	5
Facebook 686	5	8	8
Facebook 698	6	9	9
Facebook 1684	6	8	8
Facebook 1912	10	12	12
Computer Science	13	109	113
Engineering	10	63	65
Medicine	28	759	808

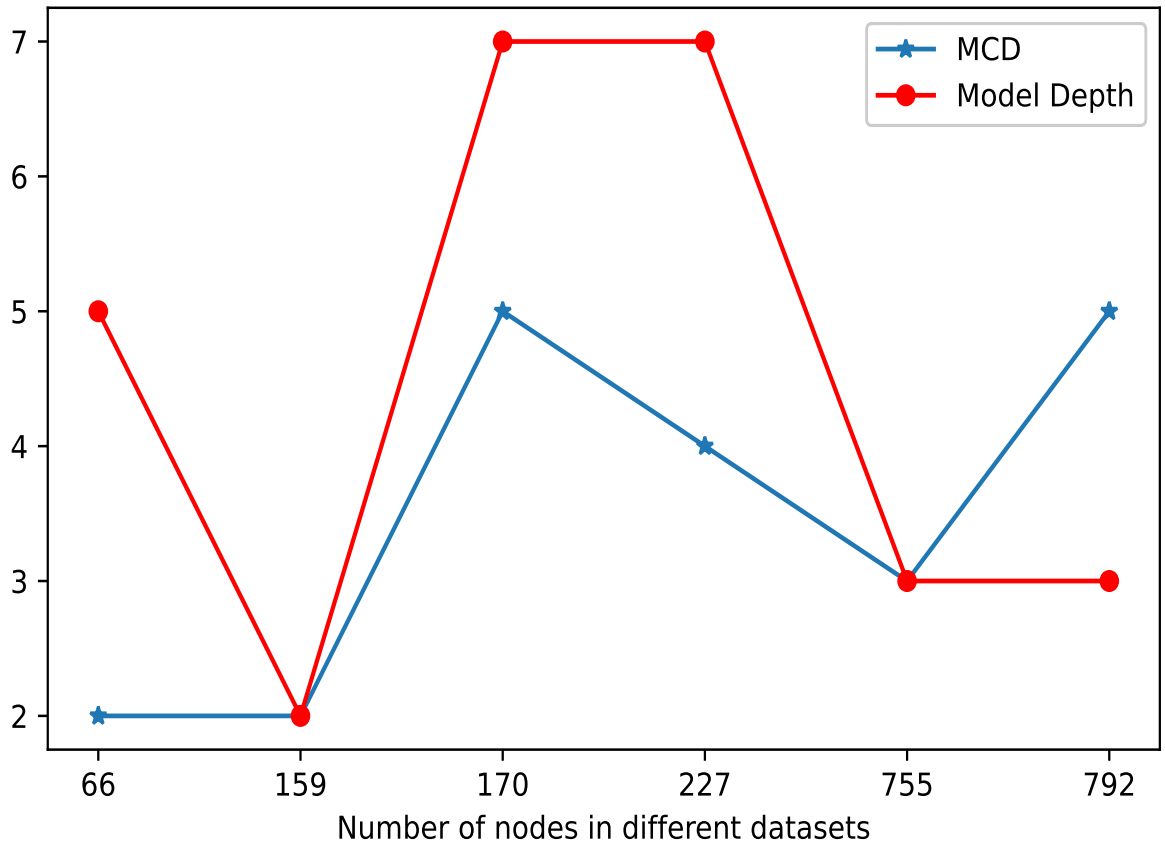


Figure 4.6: Maximum community diameter (MCD) with the best model depth.

considered variance.

4.6.3 Residual Connection

Although the dynamic dilated aggregation mechanism solves the over-smoothing problem and gives better learning capability, the deep GCN models would still fail to learn because of the vanishing gradient problem [45, 52]. It is the residual connection that provides the direct connections in the computation path of the deep neural networks. This direct computation path allows the smooth backward flow of the gradients from the last layer to the beginning layers. Consequently, the gradients do not vanish away when they reach the beginning layers. Therefore, the residual connection is essential to train deep GCN models which we use in our deep DynaResGCN model.

Another important property of residuals is the generalization capability. Some research showed that residuals increase generalization power [88]. Indeed, residual connection increases the loop in the neural network which significantly increases the complexity of the hypothesis space to be learned. A $\mathcal{O}(1/\sqrt{N})$ generalization bound is proved for neural networks with N sample size. It would be interesting to explore the generalization bounds of residual graph neural networks in terms of residual connections. In general, there is a negative correlation between the product of the norm of all weight matrices and generalization ability. In fact, very complex models do not generalize well. Therefore, it is required to use regularization of the norm of weight matrices. To enforce this idea, we also considered $L2$ regularization of the weights in our model which gives us better generalization. Most important thing is that, though it seems that residual connection makes the model complex, it does not increase any learnable parameters or weights. As a result, residual connections do not affect the generalization in a negative way.

Another interesting benefit of residual connection or skip connection is the preservation of the norm of the gradient. As it helps to preserve the norm of the gradient, it becomes possible to train deep neural networks/graph neural networks in a much more stable way with many layers [89]. Indeed, it is theoretically proved that residual connections or skip connections preserve the norm of the gradient. The more skip connections, the more norm preserving it is. Therefore, adding residual connections in our DynaResGCN model makes it much more norm preserving while considering deeper networks. However, deeper networks are more prone to over-smoothing which hampers the performance of community detection in some cases.

4.6.4 Edge Weight

The topic dataset is the only dataset where we can perform experiments on our dynamic weighted dilated aggregation algorithms. Though our weighted dilated aggregation algorithms/schemes do not outperform the random dilated aggregation algorithm, these weighted algorithms achieve

much better performance than all other considered methods. In the heatmap, we can observe a better heatmap in the case of weighted methods. Therefore, consideration of edge-weight is promising. However, it is still an open problem to consider edge-weights in a more meaningful way so that these weighted methods can outperform random dilated aggregation algorithms. In our case, we consider weights only in the case of the sub-sampling of nodes. One more meaningful way is to consider some type of weighted adjacency matrix. It is also possible to borrow ideas from some shallow models where weights of edges are considered. To the best of our knowledge, none of the works in the literature considered edge weight in the case of deep graph neural networks.

4.6.5 Network Property

Another important thing is the property of the network being studied. For example, if the network is very dense (almost fully connected) then no method can find a community structure in the network because the network has essentially no community structure. This property of a network is called the information-theoretic detection threshold [90, 91] which is well defined in the stochastic block model of community detection [92]. In general, if the graph being studied is almost regular, then there will be no community structure at all, eventually, no algorithm can detect a community structure. Therefore, when we study community detection in some graphs/networks we should also consider the network property of the detection threshold though it is well defined only for the special stochastic block model [92].

4.6.6 Evaluation Metric

The metric to determine the quality of the detected community is also crucial. In fact, using only a single metric is always highly risky because if the metric fails due to a corner case then there is no way to detect this failure. Therefore, having multiple metrics to detect different characteristics of the detected communities is of utmost importance. We alleviate this issue using multiple metrics which capture different characteristics of the detected communities. We refer to a method winning only when it performs better/good in terms of all the metrics considered.

4.6.7 Threshold Sensitivity

We have already discussed that every dataset is different from every other dataset. Therefore, the cut-off value (threshold) of the community strength (community affiliation) to determine the community belonging (whether a node belongs to a community) should be varied based on the dataset. It is not a clever idea to use the same threshold for every dataset. In our experiments, we vary the threshold in a specified range and determine the best threshold for every dataset. In the Table 4.4 and Table 4.9, we can observe the best threshold for every dataset. In these tables,

we can observe that the best threshold is not the same for all datasets. Therefore, it is necessary to consider variable thresholds in order to achieve the best result from every dataset. The key finding here is that the threshold value to determine the community belonging is subjective to the dataset being considered.

4.6.8 Robustness

From the results in Table 4.3 and Table 4.5, it is clearly evident that the DynaResGCN method is much more robust to node features than that of NOCD or other methods. In fact, the performance of DynaResGCN-G is much better than NOCD-G. When we do not have specific node features, then we consider one hot encoded feature vector for every node. As a result, the feature vector for each node is unique which helps to achieve a deeper model without over-smoothing. When we have node features, the deeper model does not work well in the case of very large graphs. The most probable reason is the over-smoothing due to similar feature vectors of many nodes. Therefore, DynaResGCN achieves better performance without node features due to lower over-smoothing and it also performs better with node features due to additional information coming from node features. These two advantages in the two different cases make the DynaResGCN method much more robust than other related methods.

4.6.9 Statistical Significance

Our best method DynaResGCN is significantly better than the nearest best method in almost every case. We measure the significance based on a standard t-test [85,86] with 95% confidence. In fact, very few works in the literature provided statistical significance for their results. Therefore, we can provably state that DynaResGCN is a significantly better method than many state-of-the-art methods for overlapping community detection.

For all the datasets having ground truth, we performed statistical tests to determine the significance of the mean difference. For each case of the results, we performed statistical significance test with 95% confidence. As we are considering 50 iterations, and each case performed random initialization, we can consider the $t-$test to measure the statistical significance of our results. If the difference is not statistically significant, then we marked both as the best. Otherwise, the best one is marked.

4.6.10 Scalability

GCN-based methods are extremely scalable for overlapping community detection. The execution time for DynaResGCN is less than 15 seconds even with 7 layers for small datasets. In this case, DynaResGCN does not impose any extra burden on the execution time. In the case of

datasets up to 22 thousand nodes, DynaResGCN took less than 2 minutes of execution time with a deep model up to 15 layers. In fact, the execution time of DynaResGCN is not more than 15 minutes even with a very large dataset ($65K$ nodes) with the deep model. Therefore, our method DynaResGCN is highly scalable even with a very deep model. As a result, the GCN-based deep methods would be much more suitable than any other method in practice where real-time overlapping community detection is required and even in the case where networks are very dynamic such as social networks.

4.6.11 Effect of Community Structure

Community structure plays a key role in the computational complexity, execution time, and inference time. For instance, if the community diameter is large, then we need a deeper model. However, if we have to detect a large number of communities, it will affect the computation of the last layer of the model. Indeed, the effect of the model depth is larger than the effect of the number of communities in terms of execution time and computational complexity. Therefore, the overall execution time is highly dependent on the community structure of the graph which is better explained by community diameter and number of communities.

4.6.12 Limitations and Future Directions

First of all, weighted dynamic dilated aggregation-based methods can not outperform the random dynamic dilated aggregation-based method. Therefore, there are scopes to incorporate information coming from edge weights in a more meaningful and sophisticated way. In the very large datasets with node attributes, the DynaResGCN-based method achieves the best performance with only 2 layers in some cases. This implies that the DynaResGCN model can not resolve the issues with over-smoothing problems perfectly. As a result, in some cases, deeper models do not outperform. Resolving the over-smoothing problem in better ways would be an interesting future research direction. In this study, we focus on the improvement of the encoder part of the whole framework. There are scopes to research on improved decoder considering different properties of community. Finally, we apply our developed methods/algorithms in a completely unsupervised setting. It would be interesting to incorporate our ideas/methods/algorithms in supervised settings of graph learning. In the case of very large $(nodes > 10K)$ and dense graphs with feature vectors for the nodes, deep models are not outperforming the shallow model with a large margin. In some cases, we get the best results in our DynaResGCN model with only two layers. Therefore, we can not state that the deeper models are always the best methods in case we have node features that may be co-related. We found that different datasets have different model depths. Therefore, we have opportunities to develop a method where model depth would be learned adaptively.

Table 4.8: Model depth of DynaResGCN-G on the different datasets

Dataset	Model Depth
Facebook 348	7
Facebook 414	2
Facebook 686	7
Facebook 698	5
Facebook 1684	3
Facebook 1912	3
Computer Science	15
Engineering	15
Medicine	15

Table 4.9: Model depth of DynaResGCN-X on the different datasets

Dataset	Model Depth
Facebook 348	7
Facebook 414	3
Facebook 686	5
Facebook 698	7
Facebook 1684	7
Facebook 1912	3
Computer Science	2
Engineering	40
Medicine	2

Chapter 5

Conclusion

Artificial intelligence can be more powerful than human intelligence in many cases. Finding patterns and statistical properties in structured data like graphs/networks are not trivial for human beings. Understanding structured data like graphs/networks is essential in many cases. Indeed, graphs/networks are ubiquitous. This physical world consists of objects. These objects are not disjoint but rather connected/related in various ways. There are a lot of efforts done by scientists in every domain to understand their specific structured data. However, these structured or related data can be represented by graphs/networks irrespective of domain. This allows us to analyze graphs/networks without getting into many details of what the objects/nodes of the graphs/networks actually represent. As a consequence, any kind of advancement in the way of analyzing/understanding/learning graph-structured data would benefit science in thousands of ways. In this thesis, we have advanced graph-structured data learning into a new stage in the case of an overlapping community detection framework.

In chapter 1, we have provided an introduction to our study. In particular, we have defined our problem of overlapping community detection framework. We have introduced our motivations and an overview of prominent applications in different areas of science and engineering. We also have included our research aim and objectives as the detection of overlapping communities in real-world networks effectively and efficiently and developing a deep graph convolutional network (GCN) which can operate on graphs having variable neighborhood sizes. Our contributions have also been discussed in chapter 1. Moreover, we have included a detailed discussion of the literature and challenges. We have discussed how the researchers attempted to develop deep graph neural networks and what are the key challenges of training deep graph convolutional networks. For example, in a regular graph, every node has the same number of neighbors. Unfortunately, in almost every case graphs/networks are irregular where every node has a different number of neighbors. As a consequence, the techniques developed by [52] to handle deeper graph neural networks are not applicable for general irregular graphs. Moreover, we also have described how some work considered graph convolutional networks in the case of community detection framework.

In chapter 2, we have explained terminologies related to this thesis and described most of the prominent works related to overlapping community detection. In particular, we have given an overview of neural networks, graph neural networks, convolutional neural networks, graph convolutional networks, shallow graph convolutional networks, etc. Among the related works, we have discussed BigClam [1], NOCD [13], DANMF [57], M-NMF [37], SLPA [21], DEMON [22], etc. We also have included all the baseline works compared in this research.

Chapter 3 describes our methodologies. In this chapter, we have introduced the concepts of dilated aggregation, dynamic edge, and edge weights successfully in the case of general irregular graphs to achieve deeper graph convolutional networks in an overlapping community detection setup. We have termed our deep graph convolutional model as Dynamic Residual Graph Convolutional Network or DynaResGCN. Moreover, we have explored different approaches to considering edge-weights of weighted graphs. These approaches combined with DynaResGCN have been described in Table 3.1. Finally, we have unified the idea of dynamic edges, dilated aggregation, and consideration of edge weights in an encoder-decoder-based framework for overlapping community detection.

Our experimentation has been described in chapter 4. We have proposed to experiment with different methods based on the different dilated aggregation schemes and different types of graphs (weighted/unweighted). In order to perform evaluation and validation, we have extended the code base of NOCD (Neural Overlapping Community Detection) [13]. We have two types of datasets: 1) without ground truth, and 2) with ground truth. The topics dataset ($n \approx 6K$) [54] is the medium-sized dataset without ground truth information. All other datasets have ground-truth information [13, 55]. However, the datasets having ground truth are of two types. Some datasets have ground-truth collected by humans through hand-labeling. In some cases, datasets are so large that hand-labeled information is not openly available in the literature. In the case of these very large datasets, empirically assigned ground truths are considered. It is clearly evident that the empirically assigned ground truth is much less reliable than the hand-labeled dataset. However, these large datasets are important to consider for scalability evaluation. For each dataset and for each of our considered methods, we have found the best hyper-parameter set through grid search. We also have performed a statistical significance test to determine the statistical significance in the case of the datasets having ground truth based on normalized mutual information (NMI).

We have described and analyzed our results in chapter 4 too. We have described how a previous study [13] demonstrated the effectiveness of graph neural networks for detecting overlapping communities where the authors showed that graph neural networks perform better than free variable optimization or other direct methods such as BigClam [1, 13], EPM [31], SNetOC, [50] etc. We have discussed how we have developed a better deep graph convolutional network-based method to detect overlapping communities with higher detection quality and much better reliability. Specifically, we have designed a deep dynamic residual graph convolutional encoder

considering dynamic dilated aggregation for the overlapping community detection model [13]. Eventually, we achieved significantly better results and detection quality compared to other related studies. In fact, our methods are much more robust to node features than the other existing methods. In almost every case, our methods achieve significantly better results than many state-of-the-art methods.

We have discussed the insights learned in this study, the limitations, and future works also in chapter 4. We have learned from experiments that deep models are important to boost the performance of the overlapping community detection model. However, there are scenarios where deep GCNs do not outperform shallow models. Therefore, we have room for improvement of deep models in the case of overlapping community detection framework. We have discussed in detail how the DynaResGCN model achieves superior performance to the NOCD model. In fact, because of the dilated aggregation in the DynaResGCN model, it can aggregate information from the second hop neighbors in one layer of the GCN without increasing the effective number of neighbors. The impact of residual connection has also been discussed in detail. We have explained/interpreted the impact of other factors including edge weight, threshold sensitivity, network property, evaluation metric, robustness, statistical significance, and scalability.

5.1 Limitations

In every research work, there are some limitations that arrive due to the scope and range of the project. Our study also has some limitations which will drive future research directions and studies. For instance, weighted dynamic dilated aggregation-based methods do not outperform the random dynamic dilated aggregation-based methods in terms of quality metrics. However, we also do not have any ground truth information for weighted datasets. As a result, our evaluation and comparison of the weighted methods are somewhat limited. We attempt to alleviate these shortcomings in the evaluation by using a novel heat-map visualization technique. Interestingly, heat-maps show that weighted methods are sometimes better than randomly considered dilated aggregation methods. As a consequence, there are scopes and opportunities to incorporate/consider the information coming from edge weights in a more meaningful and sophisticated way which will improve overall performance. There are other issues related to over-smoothing problems. For example, in very large datasets with node attributes, the DynaResGCN-based method achieves the best performance with only 2 layers in some cases. This implies that the DynaResGCN model can not resolve the issues with the over-smoothing problems perfectly. As a result, in some of the cases, deeper models do not outperform. Resolving the over-smoothing problem in better ways would be an interesting future research direction. In this study, we focus on the improvement of the encoder part of the whole framework. There are scopes to research on improved decoder considering different properties of community. Finally, we apply our developed methods/algorithms in a completely unsupervised setting. It would be

interesting to incorporate our ideas/methods/algorithms in supervised settings of graph learning. One such use case can be protein function prediction from the 3D structure. In fact, the 3D structure of proteins can be represented through 2D contact maps from which it is possible to get a contact graph where graph neural networks can operate. Thus our developed methods may help in bioinformatics and biomedical engineering. In the case of very large $(nodes > 10K)$ and dense graphs with feature vectors for the nodes, deep models are not outperforming the shallow model with a large margin. In some cases, we get the best results in our DynaResGCN model with only two layers. Therefore, we can not state that the deeper models are always the best methods in case we have node features that may be co-related.

5.2 Future Directions

In the following, we propose some open problems related to overlapping community detection and graph neural networks. These future directions actually arise from the limitations and scopes of our work.

- It is found that model depth is subjective to the dataset which is determined by hyperparameter tuning. Therefore, it is an open direction to consider a deep graph neural network where model depth would be adaptive based on the dataset.
- Provide theoretical guarantees of the capability/power of graph neural networks in the case of overlapping community detection. How much of the detection threshold DynaResGCNs can achieve in special cases like Stochastic Block Models.
- Find the theoretical limit of DynaResGCN in resolving the over-smoothing problem.
- Explore better ways of resolving over-smoothing problems.
- How to model community detection problems in the case of weighted graphs and how to evaluate and understand communities in the context of weighted graphs where both nodes and edges have weights.
- How to rethink learning of structured data like graphs in a different framework of message passing. Design/Develop/Invent a framework based on subgraph kernels or patches/filters similar to CNN to learn global structure in a better way.
- Quantum computer consists of a network of qubits. Therefore, a graph neural network seems to be a natural choice as a viable learning model in quantum computers. The research question is how to incorporate graph neural networks in the case of learning on quantum computers. Some interesting options include considering Variational Quantum Autoencoders to learn the parameters of graph neural networks.

- How to perform training/optimization/prediction in the case of very large graphs in a distributed setup like a federated learning setup where the whole graph is so large that it is not possible to store on a single device or the whole graph is inherently distributed. Moreover, it is also important to consider distributed learning on graphs to get the benefit of parallelization.
- Online learning in graphs is also another interesting research direction. In many areas like social networks, graphs are dynamic where online learning would help a lot.
- Invent/Develop/Design better global readout operation in GNNs which would not destroy the distribution of node-specific features.

References

[1] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 587–596, 2013.

[2] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 1393–1402, 2018.

[3] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[4] J. S. Coleman *et al.*, "Introduction to mathematical sociology.," *Introduction to mathematical sociology.*, 1964.

[5] B. Wellman, "The development of social network analysis: A study in the sociology of science," *Contemporary Sociology*, vol. 37, no. 3, p. 221, 2008.

[6] M. J. Herskovits, "Cultural anthropology," 1955.

[7] J. Moody and D. R. White, "Structural cohesion and embeddedness: A hierarchical concept of social groups," *American sociological review*, pp. 103–127, 2003.

[8] J. A. Chan-Lau, "Systemic centrality and systemic communities in financial networks," *Quantitative Finance and Economics*, vol. 2, no. 2, pp. 468–496, 2018.

[9] A. W. Rives and T. Galitski, "Modular organization of cellular networks," *Proceedings of the national Academy of sciences*, vol. 100, no. 3, pp. 1128–1133, 2003.

[10] V. Spirin and L. A. Mirny, "Protein complexes and functional modules in molecular networks," *Proceedings of the national Academy of sciences*, vol. 100, no. 21, pp. 12123–12128, 2003.

[11] J. Chen and B. Yuan, "Detecting functional modules in the yeast protein–protein interaction network," *Bioinformatics*, vol. 22, no. 18, pp. 2283–2290, 2006.

[12] J. Yang and J. Leskovec, "Structure and overlaps of ground-truth communities in networks," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 2, pp. 1–35, 2014.

[13] O. Shchur and S. Günnemann, "Overlapping community detection with graph neural networks," *Deep Learning on Graphs Workshop, KDD*, 2019.

[14] L. M. Sanchez-Rodriguez, Y. Iturria-Medina, P. Mouches, and R. C. Sotero, "Detecting brain network communities: considering the role of information flow and its different temporal scales," *NeuroImage*, vol. 225, p. 117431, 2021.

[15] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 021001, 2020.

[16] E. Abbe, "Community detection and stochastic block models: recent developments," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6446–6531, 2017.

[17] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[18] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller, "Verse: Versatile graph embeddings from similarity measures," in *Proceedings of the 2018 World Wide Web Conference*, pp. 539–548, 2018.

[19] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 377–386, 2017.

[20] A. Mahabadi and M. Hosseini, "Slpa-based parallel overlapping community detection approach in large complex social networks," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 6567–6598, 2021.

[21] J. Xie, B. K. Szymanski, and X. Liu, "Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process," in *2011 ieee 11th international conference on data mining workshops*, pp. 344–349, IEEE, 2011.

[22] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "Uncovering hierarchical and overlapping communities with a local-first approach," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 1, pp. 1–27, 2014.

[23] K. Asmi, D. Lotfi, and A. Abarda, "The greedy coupled-seeds expansion method for the overlapping community detection in social networks," *Computing*, pp. 1–19, 2021.

[24] Y. Gao, X. Yu, and H. Zhang, "Overlapping community detection by constrained personalized pagerank," *Expert Systems with Applications*, vol. 173, p. 114682, 2021.

[25] H. Ma, H. Yang, K. Zhou, L. Zhang, and X. Zhang, "A local-to-global scheme-based multi-objective evolutionary algorithm for overlapping community detection on large-scale complex networks," *Neural Computing and Applications*, vol. 33, no. 10, pp. 5135–5149, 2021.

[26] Y. Ruan, D. Fuhry, and S. Parthasarathy, "Efficient community detection in large networks using content and links," in *Proceedings of the 22nd international conference on World Wide Web*, pp. 1089–1098, 2013.

[27] E. Galbrun, A. Gionis, and N. Tatti, "Overlapping community detection in labeled graphs," *Data Mining and Knowledge Discovery*, vol. 28, no. 5-6, pp. 1586–1610, 2014.

[28] D. F. Gleich and C. Seshadhri, "Vertex neighborhoods, low conductance cuts, and good seeds for local community methods," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 597–605, 2012.

[29] Y. Li, K. He, D. Bindel, and J. E. Hopcroft, "Uncovering the small community structure in large networks: A local spectral approach," in *Proceedings of the 24th international conference on world wide web*, pp. 658–668, 2015.

[30] Y. Wang, Z. Bu, H. Yang, H.-J. Li, and J. Cao, "An effective and scalable overlapping community detection approach: Integrating social identity model and game theory," *Applied Mathematics and Computation*, vol. 390, p. 125601, 2021.

[31] M. Zhou, "Infinite edge partition models for overlapping community detection and link prediction," in *Artificial intelligence and statistics*, pp. 1135–1143, 2015.

[32] A. Todeschini, X. Miscouridou, F. Caron, *et al.*, "Exchangeable random measures for sparse and modular graphs with overlapping communities," *Journal of the Royal Statistical Society Series B*, vol. 82, no. 2, pp. 487–520, 2020.

[33] P. Latouche, E. Birmelé, C. Ambroise, *et al.*, "Overlapping stochastic block models with application to the french political blogosphere," *The Annals of Applied Statistics*, vol. 5, no. 1, pp. 309–336, 2011.

[34] D. Kuang, C. Ding, and H. Park, "Symmetric nonnegative matrix factorization for graph clustering," in *Proceedings of the 2012 SIAM international conference on data mining*, pp. 106–117, SIAM, 2012.

[35] Y. Li, C. Sha, X. Huang, and Y. Zhang, "Community detection in attributed graphs: An embedding approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[36] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding, "Community discovery using nonnegative matrix factorization," *Data Mining and Knowledge Discovery*, vol. 22, no. 3, pp. 493–521, 2011.

[37] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[38] J. Cao, D. Jin, L. Yang, and J. Dang, "Incorporating network structure with node contents for community detection on large networks using deep learning," *Neurocomputing*, vol. 297, pp. 71–81, 2018.

[39] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning.," in *IJCAI*, vol. 16, pp. 2252–2258, 2016.

[40] Y. Jia, Q. Zhang, W. Zhang, and X. Wang, "Communitygan: Community detection with generative adversarial nets," in *The World Wide Web Conference*, pp. 784–794, 2019.

[41] J. Chen, Z. Gong, J. Mo, W. Wang, C. Wang, X. Dong, W. Liu, and K. Wu, "Self-training enhanced: Network embedding and overlapping community detection with adversarial learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[42] C. Hu, P. Rai, and L. Carin, "Deep generative models for relational data with side information," in *International Conference on Machine Learning*, pp. 1578–1586, 2017.

[43] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, pp. 1024–1034, 2017.

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

[45] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[46] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[47] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[48] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[49] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," *arXiv preprint arXiv:1707.03815*, 2017.

[50] A. Todeschini, X. Miscouridou, and F. Caron, "Exchangeable random measures for sparse and modular graphs with overlapping communities," *arXiv preprint arXiv:1602.02114*, 2016.

[51] A. K. Ghoshal and N. Das, "On diameter based community structure identification in networks," in *Proceedings of the 18th International Conference on Distributed Computing and Networking*, pp. 1–6, 2017.

[52] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9267–9276, 2019.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[54] R. Burd, K. A. Espy, M. I. Hossain, S. Kobourov, N. Merchant, and H. Purchase, "Gram: global research activity map," in *Proceedings of the 2018 International Conference on Advanced Visual Interfaces*, pp. 1–9, 2018.

[55] J. Mcauley and J. Leskovec, "Discovering social circles in ego networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 1, pp. 1–28, 2014.

[56] D. Malhotra and A. Chug, "A modified label propagation algorithm for community detection in attributed networks," *International Journal of Information Management Data Insights*, vol. 1, no. 2, p. 100030, 2021.

[57] A. F. McDaid, D. Greene, and N. Hurley, "Normalized mutual information to evaluate overlapping community finding algorithms," *arXiv preprint arXiv:1110.2515*, 2011.

[58] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[59] "Understanding convolution operations in cnn." https://serokell.io/blog/introduction-to-convolutional-neural-networks.

[60] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[61] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[62] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[63] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[64] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 101–115, 2017.

[65] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, pp. 2224–2232, 2015.

[66] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[67] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.

[68] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[69] C.-J. Hsieh and I. S. Dhillon, "Fast coordinate descent methods with variable selection for non-negative matrix factorization," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1064–1072, 2011.

[70] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[71] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[72] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[73] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *2013 IEEE 13th International Conference on Data Mining*, pp. 1151–1156, IEEE, 2013.

[74] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics reports*, vol. 659, pp. 1–44, 2016.

[75] G. Li, M. Müller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," *arXiv preprint arXiv:2106.07476*, 2021.

[76] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[77] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*, pp. 286–297, Springer, 1990.

[78] M. J. Shensa, "The discrete wavelet transform: wedding the a trous and mallat algorithms," *IEEE Transactions on signal processing*, vol. 40, no. 10, pp. 2464–2482, 1992.

[79] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized generative models for 3d point clouds via graph convolution," in *International conference on learning representations*, 2018.

[80] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702, 2017.

[81] "Cdlib: Community discovery library in python." https://cdlib.readthedocs.io/en/latest/.

[82] J. J. Whang, I. S. Dhillon, and D. F. Gleich, "Non-exhaustive, overlapping k-means," in *Proceedings of the 2015 SIAM international conference on data mining*, pp. 936–944, SIAM, 2015.

[83] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proceedings of the 19th international conference on World wide web*, pp. 631–640, 2010.

[84] L. M. Collins and C. W. Dent, "Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions," *Multivariate behavioral research*, vol. 23, no. 2, pp. 231–242, 1988.

[85] "Comparison of two means." http://www.stat.yale.edu/Courses/1997-98/101/meancomp.htm.

[86] C. Ismay and A. Y. Kim, "Statistical inference via data science: A moderndive into r and the tidyverse." https://ismayc.github.io/moderndiver-book/B-appendixB.html#two-means-independent-samples.

[87] "Cyverse: The open science workspace for collaborative data-driven discovery." https://cyverse.org/.

[88] F. He, T. Liu, and D. Tao, "Why resnet works? residuals generalize," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 12, pp. 5349–5362, 2020.

[89] A. Zaeemzadeh, N. Rahnavard, and M. Shah, "Norm-preservation: Why residual networks can become extremely deep?," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 3980–3990, 2020.

[90] Z. Chen, X. Li, and J. Bruna, "Supervised community detection with line graph neural networks," *arXiv preprint arXiv:1705.08415*, 2017.

[91] J. Banks, C. Moore, J. Neeman, and P. Netrapalli, "Information-theoretic thresholds for community detection in sparse networks," in *Conference on Learning Theory*, pp. 383–416, PMLR, 2016.

[92] E. Abbe, "Community detection and stochastic block models: recent developments," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6446–6531, 2017.