University of Mumbai
मुंबई विद्यापीठ

We,
the Chancellor, Vice-Chancellor
and
Members of the Management Council
confer the Degree of
Bachelor of Engineering
(in its Information Technology Branch)
on
Nerurkar Pranav Ajeet Rajashree
Sardar Patel Institute of Technology
in the **First Class With Distinction**
for the examination held in April 2012
at the Convocation
held on 30th December, 2012.

आम्ही,
कुलपती, कुलगुरू
आणि
व्यवस्थापन परिषदेचे सदस्य
अभियांत्रिकी स्नातक
(माहिती तंत्रविद्या शाखा)
ही पदवी
एप्रिल २०१२ मधील परीक्षेत
प्रथम श्रेणीत विशेष प्राविण्यासह उत्तीर्ण झाल्याबद्दल
नेरुरकर प्रणव अजीत राजश्री
सरदार पटेल इन्स्टिटयूट ऑफ टेक्नॉलॉजी
यांना
३० डिसेंबर, २०१२ च्या
दीक्षान्त समारंभात प्रदान करीत आहोत.

BENG-12A-735-12112-10732

Vice-Chancellor / कुलगुरू

2535187

University of Mumbai

मुंबई विद्यापीठ

We,

the Chancellor, Vice-Chancellor
and
Members of the Management Council
confer the Degree of

Master of Engineering

(Computer Engineering Branch)

on

NERURKAR PRANAV AJEET RAJESHREE

of Thadomal Shahani Engineering College

with a Cumulative Grade Performance Index of 9.23
for the examination held in JULY 2015
at the Convocation
held on 14th January, 2016.

आम्ही,

कुलपती, कुलगुरु
आणि
व्यवस्थापन परिषदेचे सदस्य
अभियांत्रिकी अधिस्नातक
(संगणक अभियांत्रिकी शाखा)
ही पदवी
नेरुरकर प्रणव अजित राजेश्री
थाडोमल शहानी इंजिनिअरिंग कॉलेज
यांना
सचंयी श्रेणी संपादीत निर्देशांक **9.23** प्रमाणे
जुलै २०१५ मधील परीक्षेत
उत्तीर्ण झाल्याबद्दल
१४ जानेवारी, २०१६ च्या
दीक्षान्त समारंभात प्रदान करीत आहोत.

संजय वसंत देशमुख
Vice-Chancellor/कुलगुरु

15-MEAG-151-238-00000001

15163349

VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE

(Autonomous)

VJTI

Affiliated to University of Mumbai

Semester Grade Report A 16580

Student Name	Reg. No	Sem	Year
NERURKAR PRANAV	169070001	I	2016 -- 2017

Programme : Ph.D. in Computer Engineering Department : Computer Engg. & Information Technology

Course Code	Course Name	Course Credit	Grade
CO5105T	Algorithms and Complexity	3.00	AB
CO5106P	Algorithms and Complexity Lab	1.00	BB
CO5001S	Computational Methods	4.00	AA
CO5032S	Cloud Architecture Infrastructure & Technology	4.00	AA

Remarks : Examination conducted in November 2016 (Regular)

Current Semester Performance			Cumulative Performance		
Credits	Earned Grade Points	SPI	Earned Credits	Earned Grade Points	CPI
12.00	115.00	9.58	12.00	115.00	9.58

VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE MATUNGA, MUMBAI-19

22 DEC 2016

Date

Checked by

Controller of Examinations

VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE

(Autonomous)

VJTI

Affiliated to University of Mumbai

Semester Grade Report A 26665

Student Name	Reg. No	Sem	Year
NERURKAR PRANAV	169070001	II	2016 – 2017

Programme: Ph.D. in Computer Engineering Department: Computer Engg. & Information Technology

Course Code	Course Name	Course Credit	Grade
CO50058	Research Methodologies	4.00	AA
CO6801D	Seminar	0.00	PP

Remarks:

Examination conducted in April 2017 (Regular)

Current Semester Performance			Cumulative Performance		
Credits	Earned Grade Points	SPI	Earned Credits	Earned Grade Points	CPI
4.00	40.00	10.00	16.00	155.00	9.69

13 SEP 2017
Date

Checked by

Controller of Examinations

Ph.D. Pre-synopsis Report

Investigation of Techniques for Latent Space Representation of Networks

Submitted

in partial fulfillment of

the requirements of the degree of

Doctor of Philosophy (Computer Engineering)

by

Mr. Pranav. A. Nerurkar

(169070001)

Under Supervision of

Dr. S. G. Bhirud

Dr. M. M. Chandane

Department of Computer Engineering & Information Technology

Veermata Jijabai Technological Institute, Mumbai - 400019

(Autonomous Institute Affiliated to University of Mumbai)

February 2020

Dedicated

To

My Family

Declaration of the Student

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea, data, fact and source in my submission.

I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mr. Pranav. A. Nerurkar
(Reg no : 169070001) Date: _____

CERTIFICATE

This is to certify that, **Mr. Pranav. A. Nerurkar (ID- 169070001)** , a student of **Doctor of Philosophy (Computer Engineering)** has completed the Pre-synopsis report entitled **"Investigation of Techniques for Latent Space Representation of Networks"** to our satisfaction.

Dr. S. G. Bhirud	Dr. M. M. Chandane	Dr. M. M. Chandane
Supervisor	Co-Supervisor	Head, Dept of CE & IT

Date:
Place:

Pre-synopsis report Approval for Ph.D.

The Pre-synopsis report entitled **"Investigation of Techniques for Latent Space Representation of Networks"** submitted by **Mr. Pranav. A. Nerurkar (ID- 169070001)**, is found to be satisfactory for the Degree of **Doctor of Philosophy (Computer Engineering)**.

Dr. S. G. Bhirud	Dr. M. M. Chandane	Dr. G. P. Bhole	Dr. P. N. Joshi
Supervisor	Co-Supervisor	Examiner	Examiner

Date:

Place:

Acknowledgements

I am thankful to my supervisor, Prof. S. G. Bhirud, for the continuous support in my Ph.D. research, for his patience, motivation, and encouragement. His support was instrumental in keeping me on track while giving me enough space to pursue my ideas. He has been very kind and patient while suggesting the outline of this report and correcting my doubts. His guidance facilitated me during all the time of research and writing of the report.

I would like to acknowledge my co-supervisor Prof. M. M. Chandane, for his patient guidance, encouragement and advice. He inculcated objective and original thinking in me and the drive to pursue research problems. I had been fortunate to have a supervisor who cared so much about my work and responded to my questions and queries so promptly.

Besides my advisor, I would like to thank my research progress committee: Dr. Padmaja Joshi and Dr. G. P. Bhole for their insightful comments and encouragement, which helped me to widen my research from various perspectives.

Staff members of the VJTI have been very cooperative and extended their assistance at various phases of this research, whenever I approached them. I thank Prof. Sandeep Udmale, Prof. Mahesh Shirole, Dr. B. B. Meshram, Dr. V. B. Nikam and Dr. Vijay Sambhe for their extended support. My sincere thanks go to Dr. Dhiren Patel, Director of VJTI, Mumbai who provided me an opportunity to work as a research scholar and gave me access to the laboratory and research facilities which helped me to conduct this research.

Finally, I express my sincere thanks to my family for all the support and sacrifices that they have made for me during this process.

List of Abbreviations

LSR	Latent Space Representation
NRL	Network Representation Learning
IID	Independent and Identically Distributed
SNAP	Stanford Network Analysis Project
SBM	Stochastic Block Models
LSM	Latent Space Models
GAE-FC	Graph Auto-Encoder with Fully Convolutional Layer
VAE-FC	Variational Auto-Encoder with Fully Convolutional Layer
BA	Barabasi Albert
ER	Erdős–Rényi
J-R	Jackson Roger
GN	Girvan Newman
LFR	Lancichinetti–Fortunato–Radicchi
ERGM	Exponential Random Graph Models
MCMC	Markov chain Monte Carlo
SERGM	Statistical Exponential Random Graph Model
LLE	Locally linear embedding
SNE	Stochastic neighbor embedding

SVD	Singular valued decomposition
ANN	Artificial Neural Network
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
KL	Kullback–Leibler
GCN	Graph Convolutional Network
GRU	Gated Recurrent Unit

List of Symbols

Symbols	Descriptions
$\mathbf{I}_N$	Identity matrix of dimension N
$\mathbf{g}_\theta \star \mathbf{x}$	Convolution of $\mathbf{g}_\theta$ and $\mathbf{x}$
N	Number of nodes in the graph
N^v	Number of nodes in the graph
N^e	Number of edges in the graph
$\mathcal{N}_v$	Neighborhood set of node v
$\mathbf{a}_v^t$	Vector $\mathbf{a}$ of node v at time step t
$\mathbf{h}_v$	Hidden state of node v
$\mathbf{h}_v^t$	Hidden state of node v at time step t
$\mathbf{e}_{vw}$	Features of edge from node v to w
$\mathbf{e}_k$	Features of edge with label k
$\mathbf{o}_v^t$	Output of node v
$\mathbf{W}^i, \mathbf{U}^i, \mathbf{W}^o, \mathbf{U}^o, \ldots$	Matrices for computing $\mathbf{i}, \mathbf{o}, \ldots$
$\mathbf{b}^i, \mathbf{b}^o, \ldots$	Vectors for computing $\mathbf{i}, \mathbf{o}, \ldots$
σ	The logistic sigmoid function
ρ	An alternative non-linear function
tanh	The hyperbolic tangent function
LeakyReLU	The LeakyReLU function
$\odot$	Element-wise multiplication operation
$\|\|$	Vector concatenation

Abstract

A multitude of critical real-world or synthetic systems possesses network structure: citation networks, internet, protein-protein interactions, brain connectors data and social networking websites. Extending analytical techniques to process such non-euclidean data is, therefore, an essential direction for research. However, until very recently, this domain has received comparatively low levels of attention.

There is no straight forward method to analyze network data as learning models are designed for simple euclidean data or grids. Traditionally, researchers relied on hand-engineered features such as kernel functions or graph statistics for analysis of networks. As feature engineering depended on the ability of the researchers, the results of network analysis were unpredictable. Network representations also suffered from other issues such as lack of independent data-points, computationally costly calculations for graph statistics, in-applicability of parallel or distributed algorithms and the curse of dimensionality.

To address such challenges, the focus of this dissertation is on Latent Space Representations (L.S.R.) of networks, i.e., learning low dimension vector representations of network data. L.S.R. techniques have a data-driven mechanism for learning vector embedding of network data structures. Learning vector space embedding of graph-structured is similar to mapping complex data into low-dimensional geometries. The embedding process minimizes drawbacks associated with graph-structured data.

The key contributions of this dissertation are: surveying state of the art L.S.R. techniques, performing extensive experiments on network data-sets, demonstrating the use of network science in understanding the structure

and behavior in systems and investigating techniques for enhancing the performance of existing architectures for network representation learning.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

“We will never understand complex systems unless we develop a deep understanding of the networks (graphs) behind them" - Albert Laszlo Barabasi [1].

In scientific literature, different models have been developed to generate efficient representations and visualizations of data for its analysis [2]. However, networks (graphs) provide a universal language for describing and modeling complex systems [3]. Hence, they have become ubiquitous across various scientific disciplines and can be used to represent any real or artificial systems [4], for instance, internet [5], transportation systems [6], social networking websites [7, 8, 9], biological networks etc. [10, 11]. In modern times, the task of network representation models has increased in difficulty as they have to incorporate additional information of systems such as attribute data (side information), heterogeneous entities and high dimensionality. [12, 13]. To resolve the issues of network representation models, Latent Space Representation (L.S.R.) frameworks are used [14, 15]. L.S.R. frameworks learn a complex nonlinear mapping from the original (high) dimension network data to a latent (low) dimension [16].

1.1 Overview of Network Representation Models

In graph theory, a network is represented as a tuple $G = (V, E)$ where V is a (finite) set of vertices and E is a (finite) set of edges. Each edge is either a one or two element subset of V. When a network represents a system, the entities of the system are denoted

as the nodes of the network. An edge between a pair of entities indicates an interaction or a relationship between them. For instance, if a network represents the transportation system of a country, the vertices (nodes) of this network would be the different cities of a nation. The edges of the network would denote the presence of a direct transport link between one or more cities. The transportation network could be used to capture different forms of interactions. Depending on the nature of communication, directed or undirected and weighted or unweighted edges could be used. Undirected edges would represent the presence or absence of a direct route between the cities. Weighted edges would represent the volume of traffic between cities in the transportation network. Thus, graph representations offer the flexibility to capture different aspects of systems [17].

Network representation of a system allows the application of Network science for its analysis. Network science concepts have their roots in graph theory, a fertile field of mathematics. Graph theory is concerned with proving theorems and developing algorithms that can be applied to arbitrary graphs (irrespective of what these graphs model in the real world). What distinguishes network science from graph theory is that it is empiric [1]. It is this empirical aspect that makes network analysis refreshing [18]. Network science researchers study graph representations of real-world systems to understand their properties. Theoretic study of graphs from an abstract point of view is not a part of network science.

The usefulness of network representations in the examination of complex systems is illustrated in the story of Google's origin [19]. In their seminal paper, L Page *et al.* [20] argued that identification of authoritative web-pages on the internet could be done by representing the internet as a network (Figure 1.1). The web-pages (entities) could be the nodes of the network and web-pages connected by a hyperlink could be shown as nodes linked by a directed edge. In the web-graph that is thus formed, authoritative web-pages would be those with high eigenvector centrality ranking [19]. The PageRank technique proposed by the authors for information retrieval was based on this intuition [19]. Thus, network science provided a novel approach to analyze the internet. The web-graph proved useful in increasing the efficacy of information retrieval on the internet. This analogy is used to show how a concept of graph theory

was used on a network representation model to develop an efficient search engine.

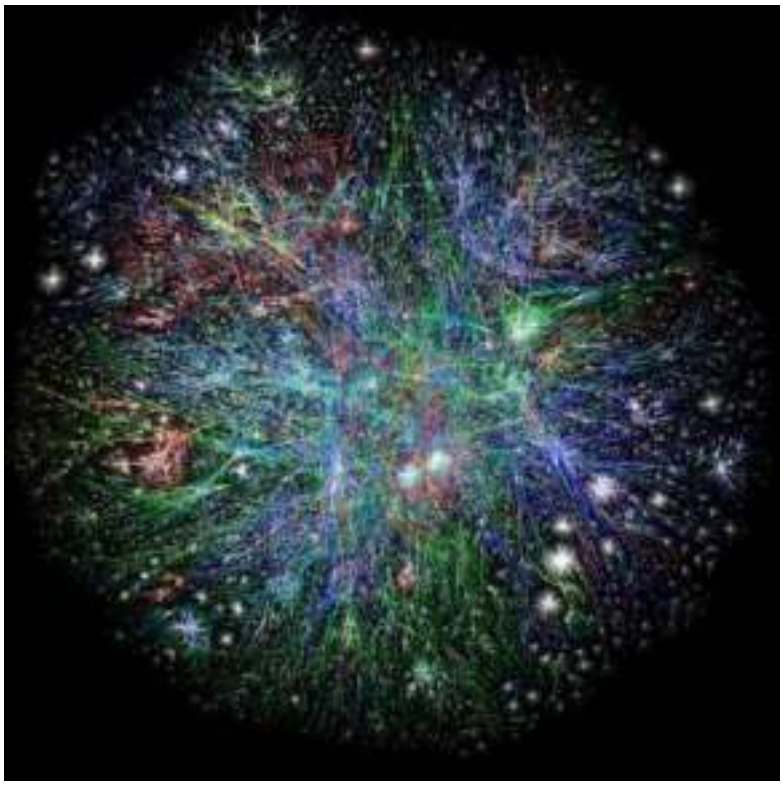


Figure 1.1: Internet network [1]

Additionally, there are several other useful statistical concepts in graph theory, which can be applied to network models to draw inferences about the nature of the systems they represent [17, 21]. Measurements of path length, diameter, connectivity, transitivity, and density allow deductions on the structural characteristics of systems. Locations of nodes in a network model can be used to draw parallels about the importance of those entities in the overall scheme of the system [18]. For instance, nodes located at the boundaries in a network are essential for information exchange with nodes outside the network and nodes situated in the center of a network are hubs which keep the network intact and play a key role in information interchange within the network.

Figure 1.2 shows a collaboration network of scientists at Santa Fe Institute (S.F.I.). Symbols indicate the research areas of the scientists and edges represent scientists that

have co-authored at least one paper. The density of edges between researchers in a particular domain is high compared to that of researchers of different domains. The position of scientists (nodes) in the Santa Fe Institute network provides insights regarding their importance to the research community of the institution. This example shows that graph theory has several concepts that can be applied to networks to understand the behavior or trends in the real-world system they represent [22].

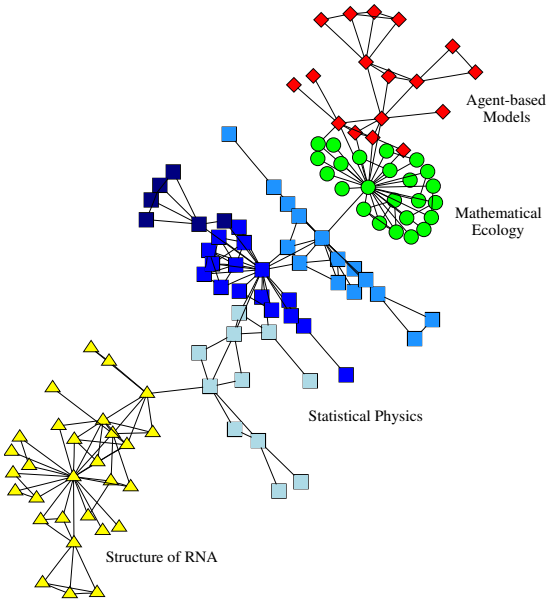


Figure 1.2: A network model representing the collaborations between scientists working at the Santa Fe Institute (S.F.I.) [18]

Similarly, data or systems across diverse domains such as computer science, transportation, social science, economics and biology too can be investigated using a network perspective.

1.2 Advantages of Representing Data as Networks

Several downstream applications were developed to obtain inferences about systems that have been represented as networks. Following sections describe some of these applications in the domains of information retrieval, machine learning and exploratory data analysis:

1.2.1 Node classification

While dealing with network representation models, certain nodes in the network are labeled. The labeling is either based on characteristics of the node or is user-defined. The objective of node classification is to assign labels to the unlabelled nodes. In Figure 1.3, each node represents a member of a club, and each edge represents a friendship between two members of that club. The data was collected from the members of a university karate club by Wayne Zachary in 1977. The network is undirected. Due to an argument between two members, the club was split into two groups, as shown in Figure 1.3. The task of node classification, in this case, is to assign appropriate group membership to unlabelled nodes represented as grey nodes in Figure 1.3.

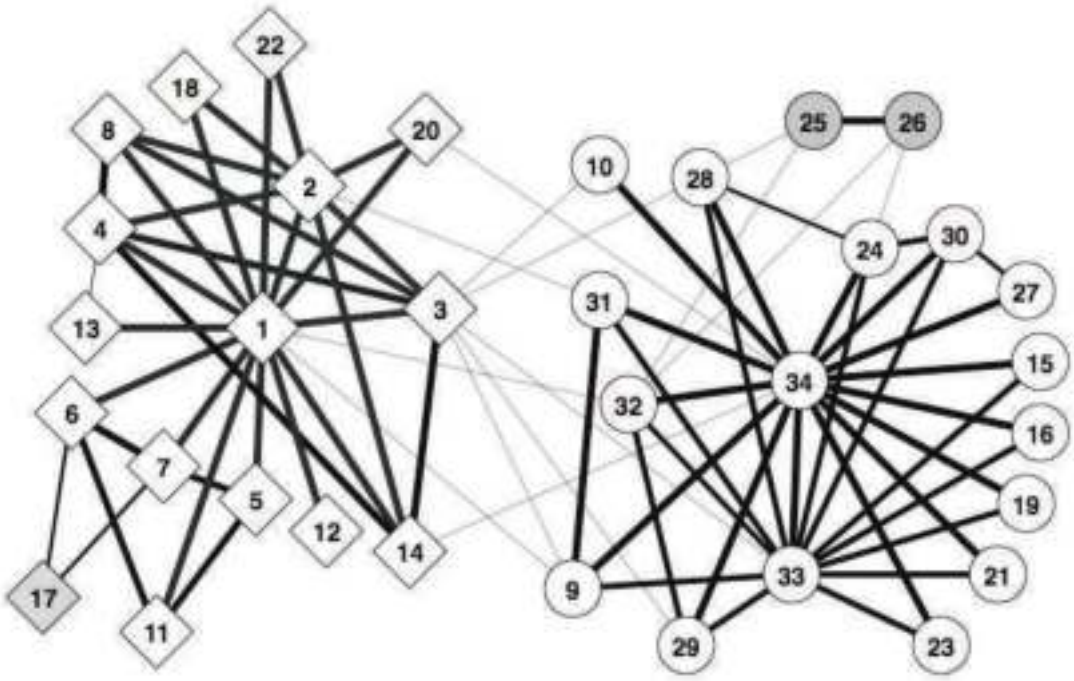


Figure 1.3: Zachary's karate club [18]

1.2.2 Node clustering

Clustering involves assigning nodes to distinct groups. In the network shown in Figure 1.4, there are three clusters of nodes. Nodes in the same cluster are at proximity with each other but distant from nodes of other clusters. Creating coarse-grained

descriptions (clustering) of data achieves effective summarization and information retrieval. As the number of clusters is not specified, the task is unsupervised and hence becomes challenging.

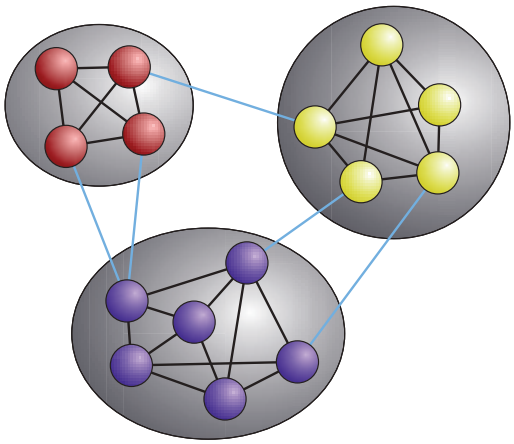


Figure 1.4: A schematic representation of a network with multiple clusters [18]

1.2.3 Link prediction

Link prediction is useful for determining future linkages in the underlying network. Such predicted linkages provide an idea of the potential relationships or missing relationships in the network (shown in Figure 1.5 with dotted red lines).

[Possible edge $(1, 2)$ depicted by dotted red line]

[Possible edge $(2, 3)$ depicted by dotted red line]

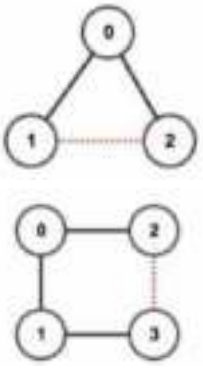


Figure 1.5: Link prediction (dashed red lines) at different networks [18]

1.2.4 Community detection

Community detection involves finding structurally related groups called communities in the network. Communities are regions of the network, which are dense in terms of the linkage behavior (shown in Figure 1.6 with dotted lines. Three of the communities share boundary vertices, indicated by the blue dots.).

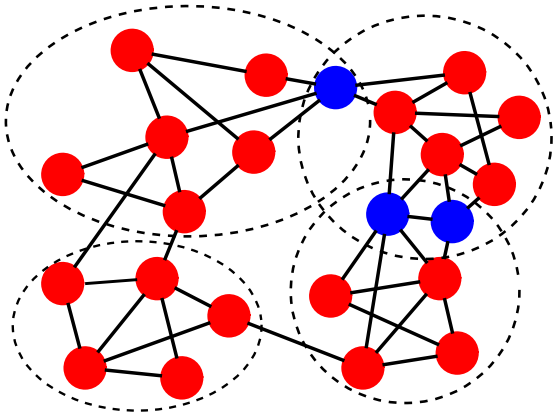


Figure 1.6: A network with four communities, indicated by the dashed contours [18]

Apart from these, there are other applications of network representation models, for instance, expert prediction and influence estimation. A standard pipeline followed for analysis of networks is shown in Figure 1.7. Raw data is a system represented in the form of a network (graph). Features of the network such as average degree, Gini index, average path length, diameter and clustering coefficient are calculated. These features are used for training a learning algorithm for network analysis.

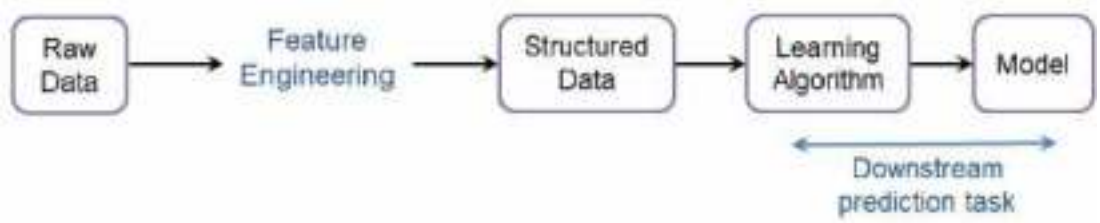


Figure 1.7: Pipeline for network analysis

As seen in Figure 1.7, feature engineering is the key aspect of the network analysis

pipeline. The performance of network analysis is determined by the quality of the features. As features vary from network to network, analysis is task-dependent. Network analysts handcraft features from the network. These features could be graph statistics or kernel functions. The efficiency of network analysis is, therefore, dependent on the skill of the researcher and the outcomes of the entire exercise are unpredictable. Task dependent feature engineering is not the only issue with network models. There are also complications when network models are used for representing complex systems. Section 1.3 describes in detail the common problems in network representation models.

1.3 Issues in Network Representation Models

As newer models and frameworks were developed for network analysis, certain drawbacks were observed due to the structural character of a graph [22, 23, 24].

1.3.1 Lack of independent data-points

For the application of machine learning on data, the underlying assumption is that the data-points should be independent and identically distributed, i.e., sampled from the same probability distribution ($i.i.d$) [25, 26]. When the data is represented as a graph (network), there is a dependency between the entities of the data. This dependency is due to the presence of edges between them. Such a dependency between data-points invalidates the $i.i.d$ assumption, and hence direct application of machine learning or probabilistic frameworks creates issues on graph data [27].

1.3.2 Curse of dimensionality

Earlier, networks were used to represent systems without any entity annotated texts, i.e., attributes (side information) [28, 29]. However, complex systems in some cases (academic networks, online social networks) are associated with attribute information. Such systems, when modeled as networks, create high dimensional data structures. Analysis of these networks is challenging using traditional algorithms as they are designed for small data-sets [30, 31].

1.3.3 Complexity in the application of parallel or distributed processing frameworks

Representing natural or synthetic systems as networks (graphs) has difficulties as strategies to investigate them depend on iterative, combinatorial operations or require computationally costly stochastic simulations [2, 27]. For instance, calculating the diameter of a graph with n nodes requires $n * n$ computations. For $n \sim 10^{3-4}$, the operation would consume several minutes. An answer to address this challenge could be the use of distributed or parallel computation models. However, the linkages between nodes (vertices) create issues in the application of parallel or distributed processing frameworks on graph data [27].

1.4 Motivation of the Dissertation

Extending techniques to process non-euclidean data such as networks is an essential direction for research. However, until very recently, this domain has received comparatively low levels of attention. Latent Space Representation (L.S.R.) emerged as the solution to address the challenges listed in Section 1.3. L.S.R. techniques learn a nonlinear mapping function from a high dimension network to a low-dimension (latent) space vector representation. There are three primary schools of thought for such techniques: Probabilistic models, Statistical models and Network Representation Learning (N.R.L.). Probabilistic models and statistical models are suitable for relatively small data-sets ($10^1 - 10^3 nodes$) and rely upon expensive and often unstable methods for probabilistic inference. Whereas N.R.L. methods apply to both small as well as large data-sets ($10^1 - 10^4 nodes$) [32, 33].

N.R.L. methods have an encoder function that learns a “similarity" between the nodes of the network representation of a system. This “similarity" is then preserved by the encoder in the node embedding (vector representation) that is generated by it. However, there is no uniformity in defining the concept of “similarity". This has led to the creation of several N.R.L. frameworks in the literature, each with its notion of “similarity". There continues to be abundant scope to propose newer definitions of “similarity" and design encoders to learn them.

Deep learning too has found applications in N.R.L. Deep learning based frameworks [34, 35, 36, 37, 38] have achieved results close to state of the art N.R.L. techniques [39]. A drawback of deep learning architectures is the complexity involved in building deep graph neural network models. The backbone of deep graph neural networks is the graph convolution operation. Graph convolution relies on a fully connected layer. This layer increases the number of parameters of the model and slows the gradient descent process. In this dissertation architecture changes are suggested to overcome these drawbacks of Graph Convolutional Networks (G.C.N.) and its variants.

L.S.R. frameworks produce node embedding (vector representation) that overcome the disadvantages found in network representations of data mentioned in Section 1.3. Therefore, this dissertation aims to design latent space representation learning frameworks for developing vector representations of networks. The focus is on surveying state of the art L.S.R. techniques, performing extensive experiments on network datasets, demonstrating the advantage of network science in understanding the structure and behavior in systems and investigating techniques for enhancing the performance of existing architectures for network representation learning.

1.5 Problem Statement and Objectives

The adjacency matrix of network $G(V, E)$ is denoted by $A \in R^{|V|*|V|}$. If $(i, j) \in E$, $A_{ij} = 1$ denoting the edge (i, j); otherwise $A_{ij} = 0$, $a_i = [A_{i,1}, ..., A_{i,n}]$ is used to denote the i^{th} row of the adjacency matrix and $V^a \in R^{|V|*p}$ is used to denote the vertex attribute matrix if present otherwise, $V^a = \phi$. $n_i^a = [n_1^a, ..., n_p^a]$ are the attributes of node a, where p is the number of node attributes. Similarly, an edge attribute matrix $E^a \in R^{|E|*f}$ may be present.

The problem is defined as follows: Given the representation of system as a network $G = (V, E)$ and associated attributes, the aim is to map it to a latent space representation. Each node u is embedded to a low-dimensional vector space y_u by learning a complex nonlinear function $f : V, V^a \rightarrow R^d$, namely $y_v = f(v, V^a) \forall v \in V$.

It is required that $d << |V|$ and the function f preserve a “similarity" measure defined on the graph G.

Intuitively, if two nodes u and v are “similar” in graph G, their embedding y_u and y_v should be close to each other in the embedding space i.e. ${y_u}^T y_v \sim 1$. The notation $f(G) \in R^{|V|*d}$ is used for the embedding matrix of all nodes in the graph G.

1.5.1 Research Gaps

- Statistical models such as Stochastic Block Model (S.B.M.) are regarded as the "The most promising class of statistical models for expressing networks into low-dimensional geometries" [40, 41, 42, 43]. Statistical models lacks effective representation of the networks as attribute information associated with the nodes is discarded while finding LSR. [40, 44]

- S.B.M. is feasible for networks with nodes in range of 10^3 whereas the latent variable model is feasible for networks with nodes in range of $10^1 - 10^2$ [45, 46, 47]. Performance of statistical models significantly reduces when the number of nodes in the network goes beyond 10^3

- The N.R.L. frameworks in the literature preserve only one of many proximity measures defined on the graph. This could lead to node embedding that is favorable for specific applications but unsuitable for others [48, 49, 50]

- Deep learning architectures in the literature are based on Graph Convolutional Networks (G.C.N.). G.C.N. and its variants focus on information aggregation. A drawback of these models is the fully connected layer used in them increases the number of parameters. As the graph size increases, the number of parameters also increases, this leads to slower convergence of gradient descent.

- The deep graph neural network architectures in the literature are two layers deep. Therefore, these G.C.N. based architectures have a small receptive field [34, 51]. However, to improve training accuracy and reduce the loss on data, a much deeper

architecture would be required. Increasing the depth of these architectures by increasing the hidden layers causes a dramatic drop in model performance.

1.5.2 Objectives of the dissertation

- Survey state of the art Latent Space Representation (L.S.R.) techniques in the literature.
- Perform extensive experiments on network data-sets from the Stanford Network Analysis Project (SNAP) for understanding the structure and behavior of systems.
- Develop ensemble methods for L.S.R. and perform experiments on network data to compare their performance vis-a-vis existing techniques.
- Propose a convolutional implementation of G.C.N. auto-encoder architectures for L.S.R. to resolve the issues of standard graph auto-encoder architectures such as Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.).
- Increasing the depth of the graph convolutional network architecture for L.S.R. by use of residual blocks to address the problem of a small receptive field of G.C.N.

1.6 Organization of the Dissertation

The focus of this dissertation is the development of new techniques for latent space representation of networks. The methods are designed for small and large network data-sets.

The rest of the dissertation is organized as follows:

Chapter 2: Taxonomy of Techniques for Latent Space Representation of Networks.

It provides an overview of latent space representation techniques. It also gives a technical foundation upon which the remaining parts of this dissertation are built. The chapter has the following key points:

- Recent advancements in the domain of latent space representation
- Mathematical background of probabilistic, statistical and N.R.L. techniques
- Evolution of literature in latent space representation of networks Drawbacks of probabilistic, statistical and N.R.L. techniques and the need for experimenting with ensemble models for L.S.R.
- Various deep graph neural networks and their performance issues

Chapter 3: Comparative Study of Statistical Models for Latent Space Representation.

Networks representing real-world systems from different domains are analyzed using statistical network models. The outline of this chapter is:

- Generative network models Stochastic block models and Latent variable models are fit to various application scenarios
- Extensive experimentation on the network data-sets from Stanford Network Analysis Project.
- Application of concepts from graph theory to network representations of real-world systems to understand their characteristics
- Highlight the need to design newer models for L.S.R.

Chapter 4: Ensemble Model for Network Representation Learning.

Describes Ensemble N.R.L. framework that learns embeddings from an ensemble of adjacency preserving based, multi-hop distance preserving based and random walk co-occurrence preserving based N.R.L. frameworks. The outline of this chapter is:

- Discuss the approaches for designing ensemble models for unsupervised learning tasks such as L.S.R.
- Mathematical model of ensemble framework for N.R.L.

- Experimental study of Ensemble N.R.L. on standard data-sets
- Discussion of the performance of the proposed Ensemble N.R.L. on data-sets

Chapter 7.1: Exploring convolutional auto-encoders for representation learning on networks.

Presents deep auto-encoder neural network architectures for learning vector embeddings of data-points. The outline of this chapter is:

- Description of graph auto-encoder and variational graph auto-encoders
- Demonstration of equivalence between operations performed in the hidden layers of an auto-encoder and laplacian smoothing operation.
- Demonstration of the efficacy of representation vectors of the data-points obtained with auto-encoders
- Architectures of graph auto-encoder with a convolutional layer (G.A.E.-F.C.) and variational graph auto-encoder with a convolutional layer (V.A.E.-F.C.).
- Comparison between the performance of the proposed approaches vis-a-vis standard graph auto-encoders

Chapter 7.1: Investigations on Residual Graph Convolutional Networks (Residual G.C.N.) for representation learning on networks.

Investigates the working of a Graph Convolutional Networks (G.C.N.) architecture for latent space representation of networks. The outline of this chapter is:

- Description of residual neural architectures
- Description of issues in Deep learning architectures
- Design of the proposed Residual Graph Convolutional Networks (G.C.N.) architecture for L.S.R.

- Extensive experiments to understand the performance of Residual Graph Convolutional Networks (G.C.N.) architecture for L.S.R.

Chapter 7: Conclusion and Future Scope

Presents the summary of the thesis, contribution and conclusions drawn from the work, and possible directions for future work.

Chapter 2

TAXONOMY OF TECHNIQUES FOR LATENT SPACE REPRESENTATION OF NETWORKS

There are three broad categories of grouping Latent space representation techniques for networks: Probabilistic models, Statistical models, and Network representation learning models [52].

1. **Probabilistic models :** Non-statistical generative models that synthesize networks on the basis of a stochastic process [53].

2. **Statistical models :** These models try to represent networks using large number of parameters to capture properties of a specific network [54]. The parameters fit to the network are used to simulate new networks and posterior predictive check is performed on the simulated networks. This check is done to understand how well the simulated networks preserve actual network properties.

3. **Network representation learning models :** These techniques aim to embed a set of objects $x_1, .., x_n$ and pairwise connections $r(x_1, x_2)$ between them into set of vectors $x_1, x_2 \in R^d$. The distance between a pair of vectors in a latent

space encodes the pairwise relationship between them in the original space i.e. $r(x_i, x_j) \sim {x_i}^T x_j$ [55].

2.1 Latent Space Representation Techniques: Probabilistic Models

Probabilistic models assume that a network forms as a result of the behavior of the actors (nodes) involved in them. Actors can have various tendencies to develop relationships with other actors, and these relationships form a network. Each type of probabilistic model assumes a different propensity to synthesize relationships, i.e., random attachment, preferential attachment, etc.

2.1.1 Types of probabilistic models

2.1.1.1 Barabasi Albert (B.A.) - Preferential attachment

Intuition behind this model is that when new nodes enter a network they prefer to attach to popular nodes (high in-degree) over others [56, 57]. The generative process of the network initializes with a single node. Then at each time step a node is created that initiates outgoing edges to the existing nodes in the system. The probability that an existing node i is chosen by an outgoing edge is given by Eq. 2.1:

$$P[i] \propto k[i]^{\alpha} \tag{2.1}$$

- α = exponent of preferential attachment;
- $k[i]$ is the in-degree of vertex i in the current time step

Thus the probability $P(k)$ that newly created nodes link with k existing nodes decays as a power law. The graph generated using this model has a power-law distribution of degrees. However, this stochastic process assumes only a linear relation for preferential attachment [4].

2.1.1.2 Erdős–Rényi Random Graph - Random attachment

These graphs are of two types $G(n, p)$ and $G(n, m)$. $G(n, p)$ has n vertices and probability of an edge between them is constant p. $G(n, m)$ has n vertices and m edges such that m edges are chosen uniformly at random from a set of all possible edges [4, 58]. In both $G(n, p)$ and $G(n, m)$ E.R. generative models, it is assumed that the nodes decide to form edges with other nodes based on a constant probability ($G(n, p)$) or uniformly at random $G(n, m)$. Hence, no preferential attachment pattern is observed.

2.1.1.3 Preferential attachment and aging

Preferential attachment and aging is a discrete-time step model of a growing random graph. At each time step, a single node is added, and it initiates links to a node already existing in the network. The probability of a node k getting an initiated edge is given by $P[k]$ in Eq. 2.2 [4]. This model thus enriches the Barabasi Albert model.

$$P[k] = (c * k[i]^{\alpha} + a) * (d * l[i]^{\beta} + b) \tag{2.2}$$

- c, d = coefficient of degree and age.
- $k[i]$, $l[i]$ = in-degree and age of node i at current time step.
- a, b = attractiveness of node with no adjacent edge and zero age.
- α, β = preferential attachment exponent, aging exponent.

2.1.1.4 Watts - Strogatz model

A generative model which creates a structured lattice graph. Each node is connected to all nodes within its neighborhood. The lattice structure that is formed is rewired, i.e., edges are selected at random with a probability p and connected to nodes outside their immediate neighborhood. This is done without altering the number of nodes or edges. The rewiring procedure creates a “Small World" Effect, i.e., a reduction in the average path length of the graph [4].

2.1.1.5 Jackson - Rogers model

M Jackson *et al.* proposed a Network Generative model where nodes of the network are allowed to form links to other nodes using a hybrid strategy that encapsulates elements of the preferential attachment model and the Erdős–Rényi model [59]. Thus, if there are pre-existing m nodes in a network, then a new node links to $a * m$ of them chosen uniformly at random and $(1 - a) * m$ using a neighborhood search strategy (choice based links) and links to them. The hyper-parameter a is the ratio of chance-based interactions to choice based interactions.

2.1.1.6 Girvan Newman benchmark, Lancichinetti – Fortunato – Radicchi benchmark

Girvan Newman (G.N.) benchmark, Lancichinetti–Fortunato–Radicchi (L.F.R.) benchmark are graphs with 128 nodes. G.N benchmark is a graph having four communities with 32 nodes each [60, 61, 62]. There are a total of 128 nodes, with each having a degree of 16. The mixing parameter μ as given in Eq. 2.3 decides each nodes community association and each node has disjoint membership to one community [62].

$$\mu = \frac{k_o}{k_i} + k_o \tag{2.3}$$

- k_o = number of edges connecting vertices in different communities
- k_i = number of edges connected to a vertex.

Girvan Newman benchmarks [18] produce networks with the Poisson distribution. This is a drawback as real-world graphs have power-law distributed network sizes. To overcome this drawback, L.F.R. benchmarks [18, 63] were proposed that had power-law distributed vertex degrees and community sizes. L.F.R. benchmark graphs are configuration models with built-in communities that may be overlapping or disjoint. Another benchmark designed to model dynamic communities was proposed by Granell *et al.* [64]. Communities are allowed to grow, shrink, merge, and split. However, at each time step, the sub-graphs are proper communities in the probabilistic sense [18].

2.1.1.7 Forest fire network model

The Forest Fire network is a generative model where one vertex is added at a time [1]. This vertex a connects to $ambs$ vertices already present in the network, chosen uniformly at random. For each chosen vertex v, the following procedure is performed:

- Generate two random numbers that are geometrically distributed with means $\frac{p}{1-p}$ and $\frac{rp}{1-rp}$ such that p is the forward probability, r is the backward probability.

Based on these probabilities, outgoing and incoming neighbors of v are connected to a. If v has neighbors below a threshold value, then all of them are connected to a.

2.1.2 Latent space representations using probabilistic models

The parameters of probabilistic models are given in Table 2.1. These parameters are estimated from the graph $G(V, E)$ that the model is fit to.

Table 2.1: Parameters of probabilistic models

Name of probabilistic model	Parameters
Barabasi Albert (B.A.)	α, $\|V\|$
Erdős–Rényi	p, $\|V\|$, $\|E\|$
Preferential attachment and aging	α, β, $\|V\|$
Watts - Strogatz	p, $\|V\|$
J-R Model	a, m, $\|V\|$
Girvan Newman Benchmark	μ, $\|V\|$
L.F.R. Benchmark	μ, $\|V\|$
Forest Fire	p, r, $\|V\|$

After the parameters are estimated, the latent representations $z_i, z_j, \ldots \in \mathbb{R}^d$ of the nodes $x_i, x_j, \ldots$ are calculated based on Eq. 2.4.

$$P(Y|Z, X, \theta) = \prod_{i \neq j} P(y_{i,j}|z_i, z_j, x_{i,j}, \theta) \tag{2.4}$$

where,

- $Z = Z \in \mathbb{R}|V|^d$ - matrix of latent representations of nodes of the network
- X = nodes of the network
- θ = parameters of the generative model
- $P(Y|Z, X, \theta)$ = probability of network Y given Z, X, θ

A drawback of probabilistic models is that even though they are “generative", they do not generate networks that share many properties with the specific network to which they were fit. This affects the quality of the latent representations of the nodes.

2.2 Statistical Models

Unlike probabilistic models, statistical models calculate the latent representations $z_i, z_j, \ldots \in \mathbb{R}^d$ of the nodes $x_i, x_j, \ldots$ using statistical estimation techniques.

2.2.1 Types of statistical models

2.2.1.1 Stochastic blocks models

Stochastic blocks models are the natural enrichment of Erdős–Rényi random networks [65]. In these models, the probability of a link formation between actors is dependent on the characteristics of the nodes, i.e., latent or observed [42, 66]. These models could be used to capture, the probability of linking with actors of the same type than with other kinds, i.e., homophily. However, they fail to capture the probability of link formation that is independent of the characteristics of actors.

Algorithm 1: Fit Stochastic blocks models to data

Result: L.S.R. of the network

1. Load adjacency matrix Y;
2. Plot singular values of Y;
3. Choose number of latent classes (blocks) by use of Eigengap heuristic; Choose dimension of latent space d = 2;
4. Fit selected model;
5. Analyze model fit: class memberships and block edge probabilities;
6. Simulate new networks from model fit;
7. Check how well simulated networks preserve actual network properties (posterior predictive check);

2.2.1.2 Exponential Random Graph Models

E.R.G.M. is an emerging statistical technique used to identify how the characteristics of the people or organizations in a network and larger social forces can explain or predict the observed patterns or ties in the observed network. E.R.G.M. allows network data to be modeled in a way similar to least squares logistic regression but does not require observations to be independent [67, 68].

E.R.G.Ms states that any network can be expressed in the form of an exponential function of certain graph statistics s_k which could be dyads, triads, k-stars, in-stars, out-stars, triangles, etc. Eqn. 2.5 provides a mathematical formulation of E.R.G.M's [69, 70].

$$P(Y) = \frac{exp[\sum \beta s_k(Y)]}{\sum_{Y'} exp[\beta s_k(Y')]}] \tag{2.5}$$

where,

- s_k = graph statistics
- P(Y) = Probability of obtaining a particular graph Y
- β = Vector of coefficients for graph statistics
- Y' = Another realization of a graph such that $Y(|V|) = Y'(|V|)$

However, there are challenges in estimating E.R.G.Ms. The graph statistics s_k may not give a clear picture regarding their significance in the network Y. Hence, the social theory regarding the domain is also needed to understand the results and draw inferences from them. The second challenge of E.R.G.Ms is estimating Y', which is a set of all possible graphs with an equal number of nodes as Y. Thus the term in the denominator $\sum_{Y'}$ has to sum over $2^{\binom{n}{2}}$ possibilities. To avoid this computation, T. Snijders *et al.* and M. Handcock *et al.* proposed Markov chain Monte Carlo (M.C.M.C.) techniques for estimation of Y' [71, 72, 73]. S. Bhamidi *et al.* argued that for E.R.G.Ms, M.C.M.C. estimation of g' should be efficient only if links in the graph are assumed to be formed independently [74]. But as this assumption is not applicable for E.R.G.Ms, M.C.M.C. leads to incorrect estimations [33, 75, 76].

2.2.1.3 Statistical Exponential Random Graph Model

Statistical Exponential Random Graph Models [S.E.R.G.Ms] were proposed to resolve the estimation issues of E.R.G.Ms. S.E.R.G.Ms assume that all network having same graph statistics s_k are equally likely [77, 78, 79]. This reduces the exponential search space of Y' and the E.R.G.M. equation is modified to Eqn. 2.6.

$$P(Y) = \frac{exp[\beta s(Y)]}{\sum_{s_k} N(s_k) exp[\beta s_k]}] \tag{2.6}$$

where,

- N(s') = Number of networks that have a particular graph statistics
- s_k = graph statistics
- P(Y) = Probability of obtaining a particular graph

2.2.1.4 Latent variable model

Latent variable model represent data as a $n * n$ sociomatrix Y, with $y_{i,j}$ denoting an edge (relation) from node i to node j, and covariate information denoted as matrix X [80]. These models assume a conditional independence approach to edge formation. This is given by Eq. 2.7. In this approach, it is assumed that the presence or absence of an edge between two nodes is independent of other edges in the system. Probability of an edge $P(y_{i,j}$ depends on the latent positions of the two nodes in a social space z_i, z_j. These positions depend on $x_{i,j}$ and θ the vector of parameters to be estimated [2, 81, 82].

$$P(Y|Z, X, \theta) = \prod_{i \neq j} P(y_{i,j}|z_i, z_j, x_{i,j}, \theta) \tag{2.7}$$

The Distance model, a variation of the latent variable model, is a logistic regression model in which the probability of a tie between two nodes i, j is given by $P(y_{i,j}|z_i, z_j, x_{i,j}, \theta)$. This probability depends on the Euclidean distance between z_i and z_j ($\psi_{i,j} = ||z_i - z_j||_2$). The probability of a tie $P(Y|Z, X, \theta)$ in this model is given by Eq. 2.8:

$$P(y_{i,j} = 1|\psi_{i,j}) = \sigma(\psi_{i,j}) \tag{2.8}$$

The above model is symmetric, $P(i \rightarrow j) = P(j \rightarrow i)$. Symmetricity means that probability of an edge formation from i to j is same as the probability of an edge formation from j to i [83]. However, in many networks such symmetry is not achieved. The shortcomings in the above model can be removed by supposing that a node i has an associated unit-length k-dimensional vector of characteristics v_i. These characteristics can be thought of as points on a k-dimensional sphere of unit radius. The angles between vector of characteristics of two actors can be: $v_i' v_j > 0$ (tie is likely), $v_i' v_j = 0$ (neutral), and $v_i' v_j < 0$ (unlikely). To this a parameter is added for each node to allow for different levels of activity viz. $a_i > 0$ be the activity level of actor i. Then the probability of a tie from i to j is considered as depending on the magnitude of $a_i v_i' v_j$ or, equivalently, $z_i' z_j = |zj|$, where $z_i = a_i v_i$. This is the signed magnitude of the projection of z_i in the direction of z_j and can be thought of the extent to which i and j share characteristics. The probability of a tie from i to j using this model is given by Eq. 2.9:

$$logodds(y_{i,j} = 1|z_i, z_j, x_{i,j}, \alpha, \beta) = \alpha + \beta' x_{i,j} - \frac{z_i' z_j}{|z_j|} \tag{2.9}$$

Algorithm 2: Fit latent variable model to data

Result: L.S.R. of the network

1. Load adjacency matrix Y;
2. Model selection: choose dimension of latent space $d = 2$;
3. Fit selected model;
4. Analyze model fit: examine estimated positions of nodes in latent space and estimated bias;
5. Simulate new networks from model fit;
6. Check how well simulated networks preserve actual network properties (posterior predictive check);

A drawback of statistical models is that they cannot be applied to data-sets with more than $10^1 - 10^2$ nodes. This is due to the computationally costly stochastic calculations that these models rely on.

2.3 Overview of Network Representation Learning

9The three classes of N.R.L. techniques discussed in the literature are Adjacency preserving methods [24, 25, 26, 30, 31, 84, 85, 86, 87, 88], Multi-hop distance preserving methods and Random walk occurrence preserving methods [89, 90, 91, 92, 93, 94, 95, 96, 97, 98]. A mathematical background and intuition behind the state of the art network embedding models is given along with a description of the evolution of N.R.L. frameworks from proximity preserving "shallow encoder" frameworks to deep learning frameworks [28, 29, 34, 35].

2.3.1 Definitions

2.3.1.1 First-order proximity

Presence of an edge is referred to as first-order proximity between two nodes. Edge weights s_{ij} (if present) can also be referred to as first-order proximities between nodes v_i and v_j. If covariates are present then, $d(v_i, v_j)$ is additionally a measure of first-order proximity.

2.3.1.2 Second-order proximity

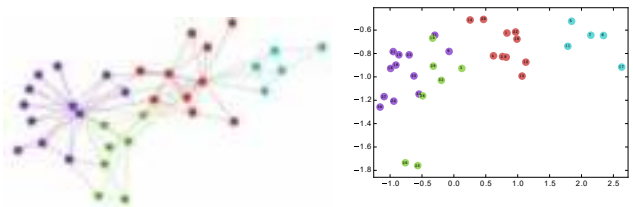
The second-order proximity between a pair of nodes describes the proximity of the pair's neighborhood structure. Let $s_i = [s_{i1}, ..., s_{in}]$ denote the first-order proximity between v_i and other nodes. Then, second-order proximity between v_i and v_j is determined by the similarity of s_i and s_j.

2.3.1.3 k^{th}-order proximity

Let $k_i = [k_{i1}, ..., k_{in}]$ denote nodes reachable by k hops from v_i. Then, k^{th}-order proximity between v_i and v_j is determined by the similarity of k_i and k_j.

2.3.2 Goal of N.R.L. frameworks

Given a network $G(V, E)$ as shown in Figure 2.1 with dimensions $\mathbb{R}^{|V|*|V|*d}$, N.R.L. frameworks learns a representation of it into a low dimension (latent) space $\mathbb{R}^{|V|*k}$ where $d <<< k$. The nodes of the original network are learned into real valued vector representations $\mathbb{R}^k$.



(a) Input: karate network (b) Output: representations

Figure 2.1: Zachary's karate club social network with communities [89]

The first network in Figure 2.1 is Zachary's karate club social network. It is represented in a low dimension space such that vector representations of the nodes in the same community in the first network are located close to each other in the low dimension space.

2.3.3 Components of N.R.L. frameworks

N.R.L. frameworks consist of three components as shown in Figure 2.2: An encoder (it converts nodes of the original network into real-valued vertex representations), a similarity measure in the original network (this similarity is preserved in latent space such that nodes that are "similar" in the original network are also "similar" in latent space) and an optimization process (this trains the parameters θ of the encoder to capture the similarity measure in the latent space). The several classes of N.R.L. techniques are due to differing definitions of the concept of similarity in the original space.

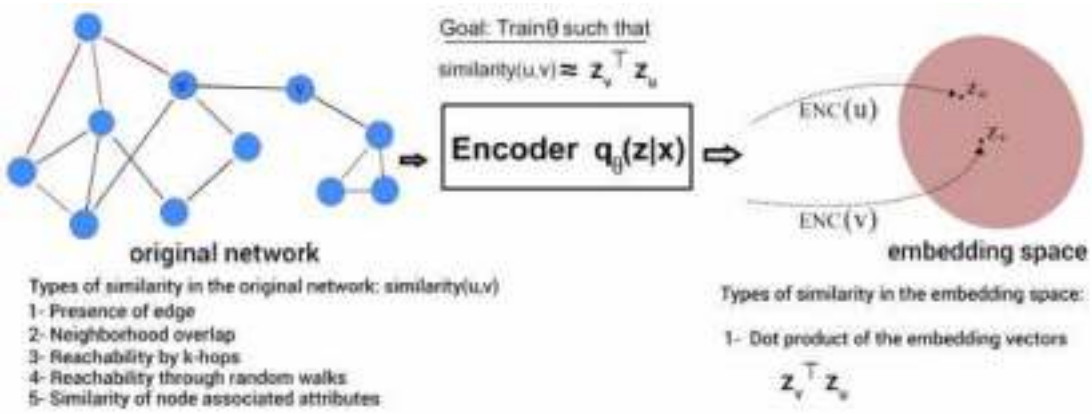


Figure 2.2: Components of N.R.L. frameworks

2.3.4 Types of N.R.L. frameworks

Adjacency preserving methods define the similarity between two nodes u, v in the original network as the presence of an edge between them in the adjacency matrix A, i.e., $A_{u,v} = 1$. Such nodes should have a dot product of the vertex representations close in the latent space, i.e., $z_v^T . z_u = 1$ when $A_{u,v} = 1$. Thus, these class of N.R.L. techniques is first-order proximity preserving. The parameters θ of the encoder are trained by an

optimization process such as Matrix factorization or gradient descent to capture this similarity and minimize the loss L as given in Eq. 2.10.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - A_{u,v}||^2 \tag{2.10}$$

Multi-hop distance preserving techniques define similarity in the original network as the existence of a k-hop path between two nodes u, v, i.e., $A^k_{u,v} = 1$ where $A^k_{u,v}$ is the k-hop adjacency matrix. Thus, these methods are k^{th}-order proximity preserving. The optimization process trains the encoder to minimize the loss function given in Eq. 2.11. In effect, the encoder learns vertex representations to preserve the k-hop distance between the nodes.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - A^k_{u,v}||^2 \tag{2.11}$$

Neighborhood overlap preserving techniques define similarity in the original network as the degree of overlap of neighborhoods N of two nodes $N_u \cap N_v$. Using an optimization process, the encoder is trained to minimize the loss function given in Eq. 2.15. Thus, the encoder learns vertex representations to preserve the neighborhood overlap matrix or second-order proximity between nodes.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - N_{u,v}||^2 \tag{2.12}$$

Random walk occurrence preserving techniques define similarity in the original network as the probability of reaching a node v by a random walk from node u, i.e., $P(v|u)$. The encoder learns vertex representations to preserve the random walk co-occurrence probability between the nodes as given by Eq. 2.13.

$$L = \sum_{(u\in V)} \sum_{(v\in N_r(u))} -log(P(v|z_u)) \tag{2.13}$$

2.4 Categories of Network Representation Learning Methods

2.4.1 Adjacency preserving techniques

The first step of Isomap is determining the neighbors of each point. It constructs a neighborhood graph and uses this graph to compute the shortest path between two nodes. Nodes having the shortest path are considered as “similar" in the original high-dimension space. Isomap utilizes this nearest neighborhood similarity matrix for obtaining the embedding matrix Z. The critical problem of Isomap is its high computational complexity due to the computation of pair-wise shortest paths. Locally linear embedding (L.L.E.) eliminates the need to estimate the pairwise distances between vertices. L.L.E. transforms data into an affinity graph based on the feature vectors of nodes, before applying gradient descent to train embedding [48, 99, 100]. Other adjacency based methods are Landmark classical scaling [49], Stochastic neighbor embedding (S.N.E.), Singular valued decomposition and Matrix Factorization [31]. The critical drawback of adjacency-based techniques is that they generate node embedding that over-fit the adjacency matrix of the original graph. This causes failure in detecting inconspicuous connections, and hence, this category of techniques cannot be used for downstream network applications like link prediction. However, for applications such as graph reconstruction, adjacency-based methods are preferred. Other drawbacks are $O(|V|^2)$ run-time ($|V|^2$ node pairs have to be considered), $O(|V|)$ parameters and consideration of only direct, local connections (the tendency of preserving only first-order proximity between nodes).

2.4.2 Multi-hop distance preserving techniques

S Cao *et al.* proposed GraRep which accepts as input the adjacency matrix A of the graph, Maximum transition step (hops) K, Log shifted factor β and dimension of representation vector d. For each transition step, the k-step log probabilistic matrix is computed $A^k_{i,j}$ as given in Eq. 2.14. This matrix is used as an approximation for the k-hop similarity matrix.

$$A^k_{i,j} = max(log(\frac{(A_{i,j}/d_i)}{\sum_{l \in V}(A_{l,j}/d_l)^k})^k - \alpha, 0) \tag{2.14}$$

The singular valued decomposition of this matrix gives W^k, which has a representation of current vertices as column vectors. Concatenation of W^k for all k's gives the final graph representations [93]. J Tang *et al.* proposed Large-scale Information Network Embedding (L.I.N.E.), which is suitable for multiple types of information networks: undirected, directed and weighted. It preserves first-order proximity ($k = 1$), i.e., closeness by distance using tie strength as well as second-order proximity ($k = 2$), i.e., global level proximity using shared neighborhood. The authors argue that the objective function preserves both the local and global network structures. An edge-sampling algorithm is proposed instead of classical stochastic gradient descent to achieve optimization. [24].

The second category of multi-hop based techniques use k-hop neighborhood overlap as a measure of similarity. Measures such as Adamic-Adar score or Jaccard similarity are used to construct a neighborhood similarity matrix S as given in Eq. 2.15.

$$L = \sum_{(u,v) \in (V*V)} ||z_u^T z_v - S_{u,v}||^2 \tag{2.15}$$

M Ou *et al.* proposed High-Order Proximity preserved Embedding (H.O.P.E.) for preserving asymmetric transitivity in directed graphs is one such technique. It uses a neighborhood similarity matrix S calculated by Jaccard similarity and trains embedding that capture this similarity in low dimension space. The objective function of H.O.P.E. is given in Eq. 2.16. It is the L_2-norm of the loss function. HOPE utilizes singular valued decomposition to minimize this loss function [85].

$$min||S - U^s.U^{tT}||_F^2 \tag{2.16}$$

$$S = M_g^{-1}.M_t \tag{2.17}$$

$$M_g = I - \beta.A \tag{2.18}$$

$$M_t = \beta.A \tag{2.19}$$

where,

1. S = high order proximity matrix in Eq. 4.2, Eq. 2.17
2. U^s, U^t = source and target embedding vectors
3. M_g, M_t = polynomial of adjacency matrix A in Eq. 2.18, Eq. 2.19
4. I = identity matrix, β = decay parameter of Katz index

Y Shi *et al.* demonstrated the applicability of neighborhood overlap matrix to calculate embedding for Heterogeneous information networks. The technique proposed by the authors was Embedding Learning by Aspects in Heterogeneous Information Networks (A.S.P.E.M.). The similarity between nodes in the network was defined in terms of aspects. Each aspect was defined as a unit representing one underlying semantic facet of the system. The authors argued that such an approach mitigated the incompatibility among aspects by considering each aspect separately. The applicability of this network was on graphs $G = (V, E)$ with a node type mapping $\phi : V \rightarrow T$ and an edge type mapping $\psi : E \rightarrow R$ where nodes are of multiple types $|T| > 1$ and edges $|R| > 1$ too are of multiple types [22]. Heterogeneous Preference embedding model was proposed by CM Chen *et al.* for embedding a user-entity network. The entities were song tracks, albums, singers to which the user listened to. The edge weights of the network represented the frequency of user-entity interactions. This is given in Eq. 2.20. Neighborhood overlap between two users reflected their preferences towards the entities, and the learned representations would reflect this [101].

$$Pr(v_j|\phi(v_i)) = \begin{cases} 1 & if v_j \in Context(v_i) \\ 0 & otherwise \end{cases} \tag{2.20}$$

$$O = -\sum_{i,j \in S} w_{i,j} log p(v_j|\phi(v_i)) + \lambda \sum_i ||\phi(v_i)||^2 \tag{2.21}$$

where,

- $\phi(v_i)$ = vector representation of v_i
- w indicates the weight of the edge
- S is a set of sampling pairs

- λ weight to prevent over-fitting
- $v_j \in Context(v_i)$ (in)direct connection to v_i or sequences of nodes in neighborhood
- $Pr(v_j|\phi(v_i))$ posterior probability
- O objective function optimized using stochastic gradient descent as shown in Eq. 2.21

Label informed Attributed Network Embedding (L.A.N.E.) [25] has a joint objective function based on learning embedding by utilizing two similarity matrices (S_1, S_2) as given in Eq. 2.22. S_1 is the adjacency matrix and S_2 is attribute similarity matrix calculated pairwise using Jaccard similarity.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - S^1_{u,v}||^2 + ||z_u^T z_v - S^2_{u,v}||^2 \tag{2.22}$$

Accelerated Attributed Network Embedding (A.A.N.E.) [26] utilizes a similarity matrix ATT based on attribute matching of the pair of nodes to learn embeddings as given in Eq. 2.23.

$$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - ATT_{u,v}||^2 \tag{2.23}$$

Similar to Adjacency based measures, methods of this category of techniques also have to iterate over V^2 pair of nodes. In addition to that, deterministic node similarity measures have to be hand-designed, which is the crucial drawback of multi-hop based techniques.

2.4.3 Random walk co-occurrence preserving techniques

These techniques perform random walks on the graph from different nodes in the network. A multi-set is created consisting of nodes visited on random walks. A three-layered neural network with a single hidden layer (no activation function) is used for generating embedding from the random walks. The artificial neural network (A.N.N.) is trained using one-hot-encoded (O.H.E.) node vectors. Each training pair is x, y where

x is the O.H.E. for the starting node of the random walk and $y \in R^{|V|}$ is a vector of nodes such that $y^i = 1$ if i^{th} node was present in the random walk. The final layer of the neural network is a softmax layer that outputs the probabilities of nodes co-occurring on random walks from the input node (Eq. 2.24). The weights of the A.N.N. are optimized by back-propagation to minimize the loss specified in Eq. 2.24. After the optimization, the weights of the hidden layer are used as the node embedding [102]. Use of A.N.N. to generate node embeddings in Random walk-based models is given in Figure 2.3 [102]. This A.N.N. is called Skip-gram architecture. The Random walk co-occurrence preserving techniques differ in the strategies used for random walks on the original graph.

$$L = \sum_{(u \in V)} \sum_{(v \in N_r(u))} \frac{exp(z_u^T z_v)}{\sum_{n \in V} exp(z_u^T z_v)} \tag{2.24}$$

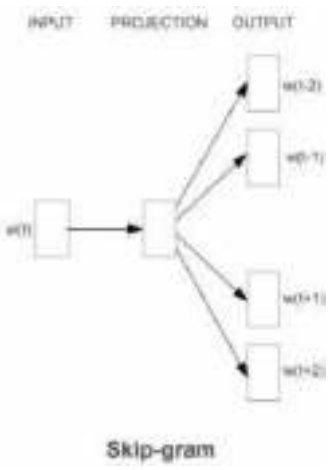


Figure 2.3: Skip-gram architecture [102]

B Perozzi *et al.* proposed WALKLETS for learning multi-scale representations of vertices in a network. The authors argue that multiscale relationships can be obtained by sub-sampling short random walks on the vertices of a graph. By 'skipping' over steps in each random walk, WALKLETS generates a corpus of vertex pairs that are reachable via paths of a fixed length. This corpus is then used to learn a series of latent representations using Skip-gram, each of which captures higher-order relationships from the adjacency matrix [23] successively. DeepWalk first generates random walks

on the network. Each walk contains many vertices in a line. Then each walk is treated as a sentence and applied to the Skip-gram in Figure 2.3. However, a drawback of DeepWalk is that it lacks a clear objective function and also it is designed only for networks with binary edges [89]. As DeepWalk uses plain random walks, it cannot capture second-order proximity of nodes, a modified strategy for running random walks was proposed in Node2vec by J Leskovec *et al.*. Random walks used in Node2Vec alternate between depth-first search (D.F.S.) and breadth-first search (B.F.S.) on the graph in a random manner. The authors speculated that such a hybrid strategy captured both first-order (using D.F.S.) and second-order (using B.F.S.) proximity. Node2vec's sampling strategy accepts four arguments: Number of random walks to be generated from each node in the graph n; the number of nodes in each walk l; P return hyper-parameter; q in-out hyperparameter, context window size and the number of iterations. The algorithm for the random walk generation will go over each node in the graph and generate n random walks, of length l. If a random walker transitions from node t to node v the probability to transition from v to any one of his neighbors is $edgeweight * \alpha$ (normalized), where α is dependent on the hyper-parameters. Using the sampling strategy, node2vec generates "sentences" (directed sub-graphs), which are multi-sets of vertices. These are then used to train the word2vec model for obtaining embeddings [90].

Gat2vec proposed by N Sheikh *et al.* decomposes the network into two data structures viz. network of connections G_s and bipartite graph of nodes and attributes G_a. It takes parameters ν_s, ν_a which are the number of random walks per node in G_s, G_a respectively, length of random walks on G_a, G_s given by λ_s, λ_a, context window c and embedding size d [92]. Random walks on both graphs are performed, and the results are provided to Skip-gram neural network to obtain the vector embeddings. C Yang *et al.* proposed Text-Associated DeepWalk for derived network embedding preserving network structure and text information associated with nodes [103]. The authors prove, the equivalence of DeepWalk and the Matrix factorization given in Eq. 2.25 (W, H are the learned embeddings).

$$\min_{W,H} ||M - W^T H||_F^2 + \frac{\lambda}{2}(||W||_F^2 + ||H||_F^2) \tag{2.25}$$

This equation is modified to include the text attribute information matrix T as given in Eq. 2.26. The vertex embeddings are formed by appending w with HT after the factorization in Eq. 2.26 is solved.

$$\min_{W,H} ||M - W^T HT||_F^2 + \frac{\lambda}{2}(||W||_F^2 + ||H||_F^2) \tag{2.26}$$

Random walk co-occurrence preserving techniques have $O(|E|)$ iterations. The performance of random walk co-occurrence preserving techniques depends on the strategy used for random walk viz. plain random walks, skipping, DFS, BFS, etc.

The other techniques belonging to these three categories of network representation learning techniques are listed in Tables 2.2 and 2.3. The approaches presented in Tables 2.2 and 2.3 require that all nodes in the graph should be present during the training of the encoder. Hence, these approaches are inherently transductive and do not naturally generalize to unseen nodes [32, 104]. The embedding matrix learned by these encoders is $Z = \mathbb{R}^{|V|*k}$. Each node of the network is present in Z as a k-dimension column vector.

Each "shallow" encoding framework generates a representation matrix $Z = \mathbb{R}^{|V|*k}$ where each column is an embedding vector for a specific node, and there is no parameter sharing between nodes. Thus "shallow" encoders require $O(|V|)$ parameters. The reason behind calling these frameworks as “shallow encoder" based frameworks are that to obtain the embedding of a node using the trained encoder a simple look-up is needed.

This, in contrast with “deep encoder" based techniques that map the nodes to their embedding vectors using a complex non-linear mapping function. As the parameters used by the “deep encoder" based frameworks for mapping the nodes to their embedding vectors are shared, these techniques are more efficient compared to “shallow encoder" based framework.

Due to the excessive parameters in “shallow encoder" N.R.L. techniques, deep neural networks were proposed. Section 2.4.4 provides a background of a Graph convolutional networks (G.C.Ns) and other deep learning techniques.

Table 2.2: Shallow encoder based network representative learning frameworks - I

Name	Optimization objective	Class of framework
Isomap [48]	Eq. 2.10	Adjacency
Locally linear embedding (LLE) [49]	Eq. 2.10	Adjacency
Landmark classical scaling [49]	Eq. 2.10	Adjacency
MF [31]	Eq. 2.10	Adjacency
LANE [25]	Eq. 2.10	Adjacency
AANE [26]	Eq. 2.10	Adjacency
HOPE [85]	Eq. 2.11	Multi-hop distance
VERSE [84]	Eq. 2.11	Multi-hop distance
HPE [105]	Eq. 2.11	Multi-hop distance
GraRep [93]	Eq. 2.15	Neighborhood overlap
LINE [24]	Eq. 2.15	Neighborhood overlap

Table 2.3: Shallow encoder based network representative learning frameworks - II

Name	Optimization objective	Class of framework
ASPEM [22]	Eq. 2.13	Random walk co-occurrence
Gat2Vec [92]	Eq. 2.13	Random walk co-occurrence
DeepWalk [89]	Eq. 2.13	Random walk co-occurrence
Walklets [23]	Eq. 2.13	Random walk co-occurrence
Struc2Vec [106]	Eq. 2.13	Random walk co-occurrence
TADW [103]	Eq. 2.13	Random walk co-occurrence
Graph2Vec [107]	Eq. 2.13	Random walk co-occurrence
Node2Vec [90]	Eq. 2.13	Random walk co-occurrence

2.4.4 Deep Learning based N.R.L. models

Graph convolutional networks (G.C.Ns) are being used to generate node embeddings based on aggregating information from local neighborhoods as given in Eq. 2.27. Every node in the network has a unique computation graph. Neighborhood aggregation can be viewed as a center-surround filter [39, 104, 108] which is mathematically related to spectral graph convolutions [34, 35].

$$h_k^v = \sigma(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}), \forall k > 0 \tag{2.27}$$

where,

- h_v^k - k^{th} layer embedding of v
- σ - Non linear activation function
- $\sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}$ - Average of neighbours previous layer embedding
- h_v^{k-1} - $k - 1^{th}$ layer embedding of v
- W_k, B_k - Trainable matrices of weights and biases of layer k

The same aggregation parameters W_k, B_k are shared for all nodes. The number of model parameters is sub-linear in $|V|$ and can be generalized to unseen nodes. The critical distinction between G.C.N. based techniques is the method of aggregating neighbor information across the layers.

The G.C.N. architecture proposed by T Kipf *et al.* uses a variation on the neighborhood aggregation approach given by Eq. 2.28. T Kipf *et al.* uses for neighborhood information aggregation [35]. Empirically, the authors found that the best results were obtained by using the same transformation matrix for self and neighbor embeddings. Compared to basic neighborhood aggregation, T Kipf *et al.* used a normalization that varied across different neighbors.

$$h_k^v = \sigma(W_k \sum_{u \in N(v) \cup u} \frac{h_u^{k-1}}{\sqrt{|N(v)||N(u)|}}) \tag{2.28}$$

W Hamilton *et al.* proposed GraphSAGE [32] that generates embedding for each node using a modified G.C.N. that relies on an aggregation function that maps two sets of vectors to a single vector as given in Eq. 2.29.

$$h_k^v = \sigma([W_k.AGG(h_u^{k-1}, \forall u \in N(v)), B_k h_v^{k-1}]) \tag{2.29}$$

Variants of GraphSAGE are Mean, Pool (Transform neighbor vectors and apply symmetric vector function) and Long Short Term Memory (L.S.T.M.) (Apply L.S.T.M. to

random permutation of neighbors). The key difference between these approaches is the type of aggregation function used in them. Eq. 2.30, 2.31 and 2.32 gives these aggregation functions [109, 110, 111].

$$AGG = \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} \tag{2.30}$$

$$AGG = \gamma(Qh_u^{k-1}, \forall u \in N(v)) \tag{2.31}$$

$$AGG = LSTM([h_u^{k-1}, \forall u \in \pi(N(v))]) \tag{2.32}$$

M Zitnik *et al.* proposed OhmNet that used graph convolutional network for constructing a link prediction system for the medicine to side-effect prediction. The heterogeneous graph of drug-protein interaction was constructed with side-effects as the edges. The architecture has an encoder to generate graph embedding and a decoder to translate embedding information for predicting side-effects [28]. OhmNet [29] is an unsupervised feature learning technique for the multi-layer network.

$$P(v_i|v_j) = \frac{exp(u_i^T * u_j)}{\sum_{i' \in A} exp(u_{i'}^T * u_j)} \tag{2.33}$$

The second order proximity between a pair of vertices v_i, v_i' is given by $p(.|v_i), P(.|v_i')$. The conditional distribution $p(.|v_i)$ is to be matched to its empirical distribution $\hat{p}(.|v_i)$ which can be achieved by minimizing Eq. 2.34 using KL divergence $d(.,.)$. λ_j is the degree of the vertex and $\hat{p}(v_i|v_j) = \frac{w_{i,j}}{deg_j}$.

$$O = \sum_{j \in B} \lambda_j d(\hat{p}(.|v_i), p(.|v_i)) \tag{2.34}$$

$$O = -\sum_{i,j \in E} w_{i,j} log p(v_j|v_i) \tag{2.35}$$

The Eq. 2.35 is optimized using stochastic gradient descent with negative sampling or edge sampling [112]. J Liang *et al.* proposed Semi-supervised Embedding in Attributed Networks with Outliers (S.E.A.N.O.) to learn a low-dimensional vector representation that systematically captures the topological proximity, attribute affinity, and label similarity of vertices in a partially labeled attributed network. The architecture

of S.E.A.N.O. consists of an Artificial Neural Network with two input layers and two output layers. Non-linear activation functions are used to transform the features into a non-linear low-dimensional space. Back-propagation and mini-batch stochastic gradient are used to minimize the objective function. A Tsitsulin *et al.* proposed VERtex Similarity Embedding (V.E.R.S.E.) a neural network-based network embedding model. Given a graph, a similarity function sim_G, and the embedding space dimensionality d, the output embedding matrix W is set to $N(0, 1/d)$. The objective function is optimized using gradient descent. This is done by repeatedly sampling a node from the positive distribution P, sample the sim_G (e.g., pick a neighboring node), and draw s negative examples. P and Q are $\sim U(1, n)$. The authors have however not given the justification for the time $O(dsn)$ and space complexity $O(n^2)$; d is the latent space dimensionality, n the number of nodes, m the number of edges, and s the number of samples used [84].

Y Li *et al.* proposed a Gated G.C.N. approach that allowed for parameter sharing across the layers of a neural network. Gated G.C.N. used neighborhood aggregation with R.N.N. state update as follows:

- Get "message" from neighbors at step k - $m_v^k = W \sum_{u \in N(v)} h_u^{k-1}$
- Update node "state" using Gated Recurrent Unit (GRU) such that new node state depends on the old state and the message from neighbors - $h_v^k = GRU(h_v^{k-1}, m_v^k)$

Figure 2.4 describes a variant of existing G.C.N. based architecture that uses two layers of feature aggregation so that second order dependencies can be captured in the vector embedding of the node. The inputs are adjacency matrix A and node feature matrix X. In the absence of the node feature matrix, the input can be matrix with one-hot encoded representations of the nodes. The output of the model are the R^d vector embeddings of the nodes of the input graph [113, 114]

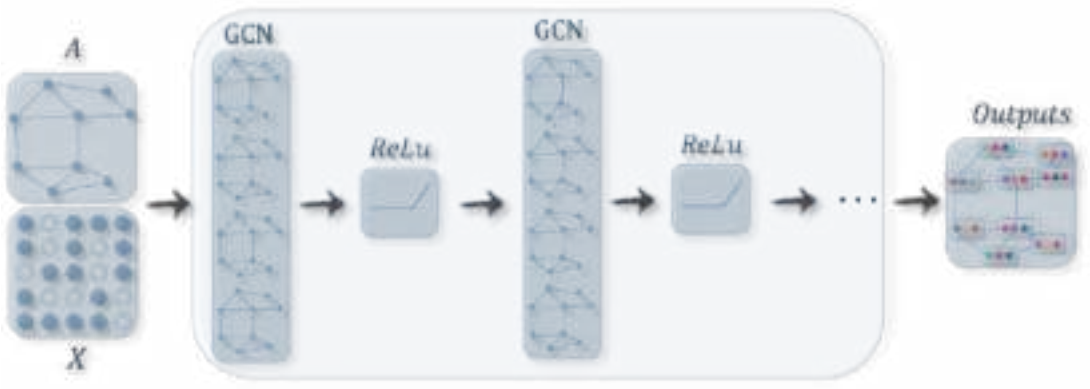


Figure 2.4: Two layered graph convolutional network [113, 114]

Instead of stacking multiple layers of G.C.N. for feature aggregation, the framework in Figure 2.5 samples node neighborhood to form sub-graphs using pooling operation. Feature aggregation is performed on the sub-graphs for generating the final hidden representations of nodes. The pooling layer is to sample the graph into sub-graphs. The node representations are calculated on the sub-graphs instead of the original graph [113, 114].

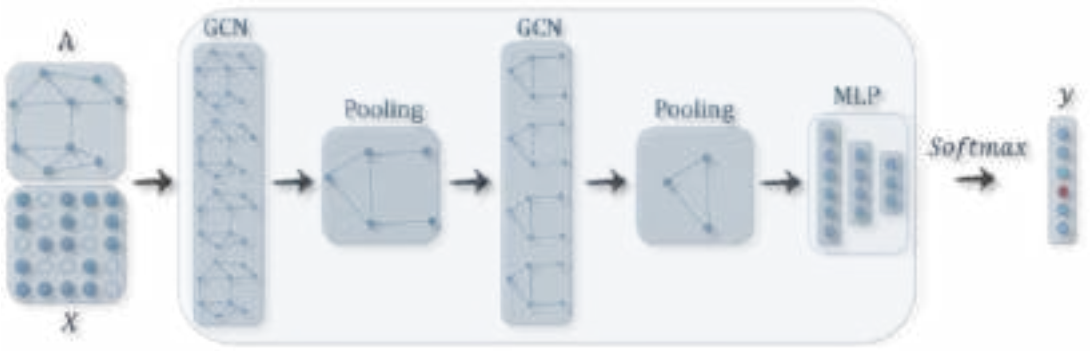


Figure 2.5: Graph convolutional networks with pooling modules [113, 114]

Graph Auto-Encoder with G.C.N. given in Figure 2.6 takes as input the adjacency matrix A of the entire graph and the node feature matrix X. It outputs the reconstructed adjacency matrix $\hat{A}$ and the node embedding matrix Z. The loss function during learning is reconstruction loss which is calculated as $L = E_{q(Z|X,A)}[logp(A|Z)]$. A single layered G.C.N. is taken for obtaining the representations $Z = GCN(X, A)$. An inner product function is used as the decoder $\hat{A} = (ZZ^T)$ [113, 114].

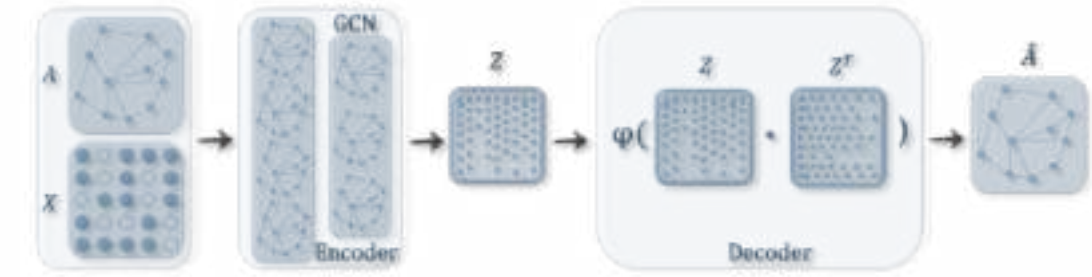


Figure 2.6: Graph auto-encoder with G.C.N. [113, 114]

Graph Spatial-Temporal Networks with G.C.N. given in Figure 2.7, uses G.C.N. to learn representations of nodes to capture spatial dependency of nodes and 1-D Convolutional Neural Network (C.N.N.) to capture temporal dependency. The framework takes as input the adjacency matrix at time t and node feature matrix at time t [113, 114].

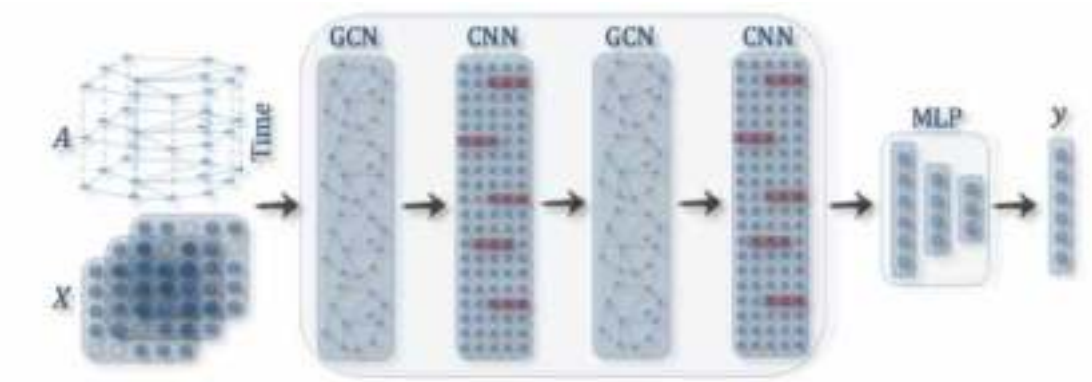


Figure 2.7: Graph spatial-temporal Networks with G.C.N. [113, 114]

G.C.N. architectures in this section are two layers deep; hence, it is a challenge to build models with multiple layers of neighborhood aggregation [115, 116]. The issues that are seen in G.C.Ns with multiple layers are excessive parameters, over-fitting and vanishing or exploding gradients during back-propagation.

Table 2.4: Different variants of graph neural networks [113, 114]

Name	Variant	Aggregator	Updater
Spectral Methods	ChebNet	$\mathbf{N}_k = \mathbf{T}_k(\tilde{\mathbf{L}})\mathbf{X}$	$\mathbf{H} = \sum_{k=0}^{K} \mathbf{N}_k \mathbf{\Theta}_k$
	1^{st}-order model	$\mathbf{N}_0 = \mathbf{X}$ $\mathbf{N}_1 = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}_0\mathbf{\Theta}_0 + \mathbf{N}_1\mathbf{\Theta}_1$
	Single parameter	$\mathbf{N} = (\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
	GCN	$\mathbf{N} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
Non-spectral Methods	C.N.N.	$\mathbf{h}^t_{\mathcal{N}_v} = \mathbf{h}^{t-1}_v + \sum_{k=1}^{\mathcal{N}_v} \mathbf{h}^{t-1}_k$	$\mathbf{h}^t_v = \sigma(\mathbf{h}^t_{\mathcal{N}_v}\mathbf{W}_L^{\mathcal{N}_v})$
	DCNN	Node classification: $\mathbf{N} = \mathbf{P}^*\mathbf{X}$ Graph classification: $\mathbf{N} = 1_N^T\mathbf{P}^*\mathbf{X}/N$	$\mathbf{H} = f(\mathbf{W}^c \odot \mathbf{N})$
	GraphSAGE	$\mathbf{h}^t_{\mathcal{N}_v} = \text{AGGREGATE}_t\left(\{\mathbf{h}^{t-1}_u, \forall u \in \mathcal{N}_v\}\right)$	$\mathbf{h}^t_v = \sigma\left(\mathbf{W}^t \cdot [\mathbf{h}^{t-1}_v \| \mathbf{h}^t_{\mathcal{N}_v}]\right)$
Graph Attention Networks	GAT	$\alpha_{vk} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_k]))}{\sum_{j\in\mathcal{N}_v}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_j]))}$ $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\sum_{k\in\mathcal{N}_v}\alpha_{vk}\mathbf{W}\mathbf{h}_k\right)$ Multi-head concatenation: $\mathbf{h}^t_{\mathcal{N}_v} =_{m=1}^{M} \sigma\left(\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$ Multi-head average: $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$	$\mathbf{h}^t_v = \mathbf{h}^t_{\mathcal{N}_v}$
Gated Graph Neural Networks	GGNN	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k + \mathbf{b}$	$\mathbf{z}^t_v = \sigma(\mathbf{W}^z\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^z\mathbf{h}^{t-1}_v)$ $\mathbf{r}^t_v = \sigma(\mathbf{W}^r\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^r\mathbf{h}^{t-1}_v)$ $\widetilde{\mathbf{h}^t_v} = \tanh(\mathbf{W}\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}(\mathbf{r}^t_v \odot \mathbf{h}^{t-1}_v))$ $\mathbf{h}^t_v = (1 - \mathbf{z}^t_v) \odot \mathbf{h}^{t-1}_v + \mathbf{z}^t_v \odot \widetilde{\mathbf{h}^t_v}$
Graph LSTM	Tree LSTM (Child sum)	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{U}^i\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma\left(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{U}^f\mathbf{h}^{t-1}_k + \mathbf{b}^f\right)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{U}^o\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{U}^u\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{k\in\mathcal{N}_v}\mathbf{f}^t_{vk} \odot \mathbf{c}^{t-1}_k$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$
	Tree LSTM (N-ary)	$\mathbf{h}^{ti}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^i_l\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{tf}_{\mathcal{N}_v k} = \sum_{l=1}^{K}\mathbf{U}^f_{kl}\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{to}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^o_l\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{tu}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^u_l\mathbf{h}^{t-1}_{vl}$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{h}^{ti}_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{h}^{tf}_{\mathcal{N}_v k} + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{h}^{to}_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{h}^{tu}_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{l=1}^{K}\mathbf{f}^t_{vl} \odot \mathbf{c}^{t-1}_{vl}$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$
	Graph LSTM	$\mathbf{h}^{ti}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^i_{m(v,k)}\mathbf{h}^{t-1}_k$ $\mathbf{h}^{to}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^o_{m(v,k)}\mathbf{h}^{t-1}_k$ $\mathbf{h}^{tu}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^u_{m(v,k)}\mathbf{h}^{t-1}_k$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{h}^{ti}_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{U}^f_{m(v,k)}\mathbf{h}^{t-1}_k + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{h}^{to}_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{h}^{tu}_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{k\in\mathcal{N}_v}\mathbf{f}^t_{vk} \odot \mathbf{c}^{t-1}_k$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$

2.5 Issues in Deep Learning Architectures

Theoretically, deep neural networks are accurate, and this accuracy increases with additional hidden layers [34]. However, practically in deep neural networks, the problem of vanishing or exploding gradients is observed. This hampers convergence [36]. As the depth of the networks increases, the number of parameters also increases ($\sim 10^6 - 10^7$), and the accuracy during optimization gets saturated. Adding additional layers in such a scenario hampers the training accuracy while increasing the training time.

2.6 Summary of L.S.R. Techniques

Survey of state of the art Latent Space Representation (L.S.R.) techniques in the literature is presented. The three categories of grouping Latent space representation techniques for networks are examined in detail. The simplicity of probabilistic models enables a rigorous theoretical analysis of model properties. However, their limited flexibility results in poor fits to data. Even though probabilistic models are "generative," they do not generate networks that share many properties with the specific network they were fit to [117]. Hence, the dissertation does not focus on utilization of probabilistic models for latent space representations of networks.

Statistical models are S.B.M., Latent variable models, E.R.G.M./p*, S.E.R.G.M. and S.U.G.M. These techniques are powerful, flexible representation models that can encode complex theories. However, these models are computationally intensive to fit data. Hence, they are limited to relatively small data-sets and rely upon expensive and often unstable methods for probabilistic inference [40]. Latent variable models are "positional analysis" models that provide a visual and interpretable spatial representation of a network. It is also possible to model assortative mixing via transitivity. Chapter 3 discusses the results of fitting statistical models to various application scenarios.

Network embedding techniques based on shallow encoders rely on the concept of “similarity" to learn embeddings of the nodes. However, there is no universally accepted notion of “similarity". This provides a scope for investigating the working of models that learn embeddings based on multiple notions of “similarity". Chapter 4 discusses

the framework for a ensemble network embedding technique.

Deep learning approaches have made an impact on representation learning on networks (N.R.L.). N.R.L. is a challenging task as complex non-linear patterns have to be identified from data in-order to created effective representations in a low dimension space. The deep learning approaches discussed in the literature are based on graph convolutional networks (G.C.N.). G.C.N. and its variants focus on information aggregation. A drawback of these models is the fully connected layer used in them increases the number of parameters. As the input graph size increases, the number of parameters also increases, this leads to slower convergence of gradient descent. Therefore in Chapter 7.1 two auto-encoder architectures are proposed which modify existing techniques like Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.) by replacing the fully connected layers with 1-D fully convolutional ones. In addition, Chapter 7.1 discusses the performance of the proposed architecture vis-a-vis existing techniques.

However, to improve training accuracy and reduce the loss of data, a much deeper architecture would be required. The deep graph neural network architectures in the literature as well as proposed in Chapter 7.1 are two layers deep. Increasing the depth of these architectures by increasing the hidden layers causes a dramatic drop in model performance. Hence, to build deeper models with many layers of neighborhood aggregation without hampering training time and accuracy, Chapter 7.1 investigates the effects of addition of a residual connection to the graph convolutional network architecture.

Chapter 3

COMPARATIVE STUDY OF STATISTICAL MODELS FOR LATENT SPACE REPRESENTATION

3.1 Introduction

Real-world systems take diverse forms, and hence, networks that model such data are complex. Such high-dimensional data often contains redundancies and regularities that can be exploited to learn more compact and tractable representations. Unlike the frameworks of data compression like Principal Component Analysis, Multi-dimension scaling - which seek only to reduce the size of data for efficient storage - latent space representation methods seek to infer an underlying geometry in the data [32, 39]. Network data has issues such as lack of independent data-points, computationally costly calculations for graph statistics, in-applicability of parallel or distributed algorithms and the curse of dimensionality. The latent space representation methods minimize drawbacks associated with graph-structured data. In the previous chapter, statistical models such as Stochastic Block Model (S.B.M.) and latent variable model were discussed. These are regarded as the "The most promising class of statistical models for expressing networks into low-dimensional geometries" [40, 41, 42, 43]. However, these

models ignore the attributes associated with the networks [40, 44]. S.B.M. is feasible for networks with nodes in range of 10^3 whereas the latent variable model is feasible for networks with nodes in range of $10^1 - 10^2$ [45, 46, 47].

3.1.1 Summary of contributions

In this chapter, generative network models S.B.M. and latent variable model are fit to various application scenarios for obtaining latent space representations. This is done in order to understand the suitability of these models. Besides, concepts from graph theory are applied to network representations of real-world systems to understand the characteristics of the networks and infer the reasons for observing such characteristics. The objective is to perform extensive experiments on network data-sets from the Stanford Network Analysis Project (SNAP) for understanding the structure and behavior of systems. A secondary objective is to highlight the drawbacks of the S.B.M. and latent variable model in obtaining L.S.R. of graph-structured data.

3.2 Definitions and Data

3.2.1 Definitions

3.2.1.1 Average clustering coefficient (c)

The Average clustering coefficient or transitivity of a network is the probability that two incident edges are completed by a third edge to form a triangle.

$$c = \frac{|\{u, v, w \in V \mid u \sim v \sim w \sim u\}|}{|\{u, v, w \in V \mid u \sim v \neq w \sim u\}|} \tag{3.1}$$

3.2.1.2 Diameter (diam(G))

Diameter of a graph G is defined as $\text{diam}(G) = \max \min d_G(x, y)$, where d is the distance function in G and the max min is taken over all vertices $x, y \in G$.

3.2.1.3 Assortativity coefficient (r)

It is positive if vertices with high in-degrees tend to connect to each, and negative otherwise. Assortativity coefficient r for edges $i = 1, 2, ..., M$ with j, k as degrees of the vertices at the ends of the i^{th} edge.

$$r = \frac{M^{-1} \sum_i j_i k_i - [M^{-1} \sum_i \frac{1}{2}(j_i + k_i)]^2}{M^{-1} \sum_i \frac{1}{2}({j_i}^2 + {k_i}^2) - [M^{-1} \sum_i \frac{1}{2}(j_i + k_i)]^2} \tag{3.2}$$

3.2.1.4 Edge density (D)

For directed graphs $D = \frac{2|E|}{|V|(|V|-1)}$ and for undirected graphs $D = \frac{|E|}{|V|(|V|-1)}$.

3.2.1.5 Gini index (G)

It takes values between zero and one, with zero denoting total equality between degrees, and one denoting the dominance of a single node. Let $d_1 \leq d_2 \leq ... \leq d_n$ be the sorted list of degrees in the network. The Gini index G is twice the area between the Lorenz curve and its main diagonal. G is defined as:

$$G = \frac{2 \sum_{i=1}^{n} i d_i}{n \sum_{i-1}^{n} d_i} - \frac{n+1}{n} \tag{3.3}$$

3.2.1.6 Average degree (d)

$$d = \frac{1}{|V|} \sum_{u \in V} d(u) \tag{3.4}$$

3.2.1.7 Average path length ($\bar{P}$)

The average number of steps along the shortest paths for all possible pairs of network nodes

$$\bar{P} = \frac{1}{n(n-1)} \sum_{i \neq j} d(v_i, v_j) \tag{3.5}$$

3.2.1.8 Variables in network data

Network data includes :

- A set of nodes (entities, objects, actors, egos, individuals) and edges (links, ties, dyads)
- Variables measured on nodes are nodal variables, and those measured on pairs of nodes (edges) are dyadic variables

3.2.1.9 Types of node attributes or side information

A binary (or dichotomous) relation takes only two values. A valued relation takes more than two values. A valued relation whose possible values are ordered is called ordinal. A valued relation whose possible values lack an order is called categorical.

3.2.2 Network data

3.2.2.1 Data-sets with no side information

These are networks $G = (V, E)$ with vertex set V and edge set E. The vertex attribute set V^a, and the edge attribute set E^a is null. Table 3.1 gives the description of the data-set crawled from online social networking websites Twitter.com (Twt-Net) and Google+ (Gplus-Net). All data-sets are publicly available at Stanford Network Analysis Platform (S.N.A.P.) - a network data repository [118]. In Twt-Net and Gplus-Net, the system that is modeled as a network is the online social networking website. The entities of the system are the users of these platforms, and they are denoted as nodes of the network. The edges of the network are "follower" relationships between the users. If a user i follows user j, then this is denoted in the network by a directed edge from node i to node j.

Table 3.1: Description of network data-sets with no side information

Description	**Twt-Net**	**Gplus-Net**
$\|V\|$	185	923
$\|E\|$	5156	39400
c	0.44	0.3
diam(G)	8	7
r	-0.19	-0.23
$\bar{P}$	2.18	2.58
D	0.201	0.05
G	0.41	0.52
d	55 (σ = 41)	85 (σ = 106)

3.2.2.2 Data-sets with binary attributes

These are networks $G = (V, E)$ with vertex set $n = |V|$ and edge set E. The vertex attribute set is $V^a = R^{n*f}$ and the edge attribute set is $E^a = \phi$, such that f is number of features for each vertex. The feature matrix of G is denoted by F. If $F_{ij} = 1$ node i has feature j; otherwise we have $F_{ij} = 0$ (binary attributes). Table 6.1 describes data-sets obtained from Blog.com (Blog-Net), Flickr.com (Flickr-Net), a protein-protein interaction network (Protein-Net), Wikipedia.com (Wiki-Net), C.O.R.A. Research Paper Classification Data-set by Andrew McCallum of University of Massachusetts Amherst (Cora-Net) and citation indexing website Citeseer.com (Cite-Net). All data-sets are publicly available at Stanford Network Analysis Platform (S.N.A.P.) [118]. Blog-Net is a directed graph of users commenting on blogs by other users on the website Blog.com. Flickr-Net is a directed graph of users following other users on Flickr.com. Protein-Net is an undirected graph of proteins interacting with other proteins. Wiki-net is a directed graph of articles citing other articles on Wikipedia.com. In Cora-Net and Cite-Net, the nodes are the academic papers. If paper i cites paper j, then this is denoted in the network by a directed edge from node i to node j.

Table 3.2: Description of network data-sets with binary attributes

Description	**Blog-Net**	**Flickr-Net**	**Protein-Net**	**Wiki-net**	**Cora-net**	**Cite-net**
$\|V\|$	5196	7575	3890	4777	2708	3312
V^a	8189	12047	50	40	1434	3704
$\|E\|$	171743	239738	37845	54810	5429	4732
c	0.08	0.1	0.09	0.43	0.14	0.13
diam(G)	~ 2	~ 2	8	4	6	8
r	-0.02	-0.23	-0.09	-0.27	0.11	0.12
$\bar{P}$	~ 2.03	~ 2.15	3.09	2.15	1.74	1.81
D	0.012	0.008	0.005	0.004	0.0002	0.0004
G	0.39	0.67	0.63	0.62	0.42	0.43
d	66.30 (σ = 54.8)	63.3 (σ = 131.52)	19.45 (σ = 34.29)	22.95 (σ = 105.92)	2.4 (σ = 2.63)	2.8 (σ = 3.41)

3.2.2.3 Data-sets with mixed attributes

These are networks $G = (V, E)$ with vertex set $n = |V|$ and edge set E and the vertex attribute set $V^a = R^{n*f}$ and the edge attribute set $E^a = R^{|E|*k}$ where f and k are the number of features for the nodes and edges in the network respectively. Table 3.3 describes data-sets such as High-Net - an undirected network of the highways in the state of Southern California, as observed in 2016. It shows the cities connected by highways. Grey-Net is an undirected sexual contact network between characters of the television show Grey's Anatomy. It gives information about actors in relationships with other actors in the series. Trade-Net is an undirected trade network of automotive electrical goods between Asia and European countries in the year 2016. It shows countries that have trade ties with each other. Bill-Net was an undirected bill co-sponsorship network in the parliament of Slovakia in 2014. It shows information about legislators co-sponsoring bills together. All data-sets are publicly available at Stanford Network Analysis Platform (S.N.A.P.) [118].

Table 3.3: Description of network data-sets with mixed attributes

Description	**High-Net**	**Grey-Net**	**Trade-Net**	**Bill-net**
$\|V\|$	205	44	99	139
Node attributes	9	17	12	4
$\|E\|$	203	46	725	471
Edge attributes	3	2	1	1
c	0.28	0	0.437	0.32
diam(G)	16	8	2	4
r	0.12	-0.22	-0.32	0.014
$\bar{P}$	6.8	3.49	2.31	3.71
D	0.018	0.04	0.11	0.043
G	0.54	0.37	0.61	0.46
d	1.97 (σ = 2.12)	2.09 (σ = 1.72)	14.64 (σ = 18.74)	6.77 (σ = 6.23)

3.3 Analysis of Networks using Statistical Models

This section provides the analysis of real-world systems given in Section 3.2.2 from a network perspective. The concepts of graph theory defined in Section 3.2.1 are used to generate insights about the structure of these systems. The systems under investigation belong to diverse scientific domains. The objective is to highlight the drawbacks of the S.B.M. and latent variable model in obtaining L.S.R. of graph-structured data. The procedure followed for the analysis is given in Algorithm 3.

Networks such as Twt-Net, Gplus-Net, Flickr-Net, Wiki-Net, Blog-Net, Grey-Net and Bill-Net are a particular type of network called "social networks". Social networks represent the sum of all professional, friendship, or family ties of the actors involved in them. Study of such systems provides an understanding of the behavior of the actors in them.

Performing downstream network tasks such as clustering on the network representations of these systems have issues due to their scale and high dimensionality. Therefore, we generate the Latent space representation (L.S.R.) of the original networks using generative statistical models such as the S.B.M. and latent variable model. This reduces

their dimensionality and makes the downstream network task amenable. As S.B.M. and latent variable model are limited to small data-sets (10^{1-2}) and also cannot handle attributes effectively, there is a motivation for designing a latent space representation model that can overcome their drawbacks.

Algorithm 3: Procedure to understand structure and behavior of networks

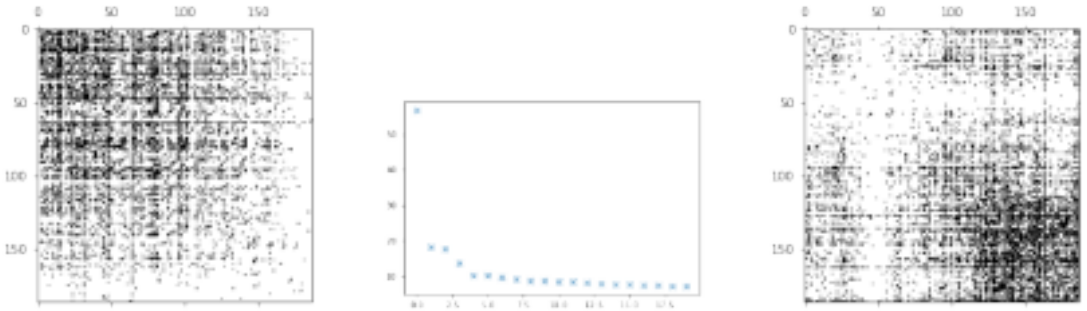
1. Load adjacency matrix Y of the network G;
2. Calculate descriptive measures $|V|$, $|E|$;
3. Calculate network statistics c, diam(G), r, $\bar{P}$, D, G, d;
4. Plot the adjacency matrix and singular values;
5. Using eigen-gap heuristic observe latent classes;
6. Assign node membership according to the latent classes;
7. Re-order nodes in adjacency matrix by class memberships and plot;
8. Fit latent variable model and S.B.M. Set dimensions of latent space = 2;
9. Analyze model fit;
10. Simulate new networks from model fit. Set number of simulations = 20;
11. Check how well simulated networks preserve actual network transitivity (posterior predictive check);

3.3.1 Twitter

Twt-Net is a data-set of 185 twitter users of a community and the "follower-following" relationships between them. The average clustering coefficient, i.e., transitivity between the members is 0.44. The high transitivity, coupled with a low average path length of 2.18 suggests that information diffusion between members could be rapid. As users prefer following popular users, social networks have negative assortativity and high inequality of degree (Gini index = 0.41). Other characteristics commonly

observed in social networks are a large number of social contacts (average degree = 55). This leads to a low average path length and diameter and high edge density.

For generating L.S.R. of this network, the latent variable model is a better choice than S.B.M. as the latter does not explicitly model transitivity. Figure 3.2 shows the results of fitting S.B.M. to Twt-Net using the procedure outlined in Algorithm 1. Figure 3.1a shows a dense adjacent matrix A but no presence of communities (latent classes) can be detected in the plot. Hence, to choose the latent classes, the plot of the singular values of A needs to be obtained.



(a) Adjacency matrix of Twitter.com data-set Twt-Net

(b) Plot of singular values of Twitter.com data-set Twt-Net

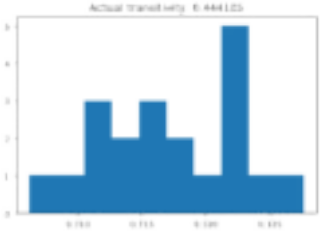
(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.1: Analysis of Twitter.com data-set Twt-Net using S.B.M.

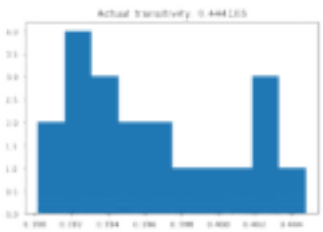
Using eigen-gap heuristic (gaps in the singular values correspond to latent classes in the network), four latent classes were observed in Figure 3.1b. The nodes in these latent classes were assigned class-memberships, and then the adjacency matrix was re-ordered. Figure 3.1c shows the re-ordered adjacency matrix with four latent classes. Once the class-memberships were assigned, the edge probabilities at the block level were calculated. Finally, new networks were simulated from edge probabilities to check the goodness of fit of the model.

Figure 3.2a shows the transitivity of regenerated networks using S.B.M. The transitivity ranges from 0.3 - 0.33 (upper and lower bounds of the 95% confidence interval), which is 0.11 lower than the actual transitivity of Twt-Net. Figure 3.2b shows the transitivity of the regenerated networks using a latent variable model. It is seen in Figure 3.2b that

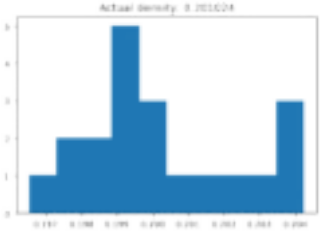
the transitivity of the regenerated networks is in the range of 0.4-0.404 (upper and lower bounds of the 95% confidence interval), which is 0.04 less than the actual transitivity of Twt-Net. Similarly, from Figure 3.2c, the edge density of the regenerated networks using the latent variable model is in the range of 0.19-0.204 (upper and lower bounds of the 95% confidence interval). Whereas, the actual edge density of Twt-Net is 0.201. Thus, the latent variable model was able to generate L.S.R. that could replicate the transitivity (as shown in Figure 3.2b) and edge density (as shown in Figure 3.2c) of the original network. However, S.B.M. has generated L.S.R. that does not regenerate the original network (Twt-Net) effectively as a latent variable model. Conclusions from the posterior predictive check of S.B.M. and latent variable model is that L.S.R. generated using latent variable model are more effective than S.B.M. in Twt-Net.



(a) Simulate new networks from Stochastic Block Models (S.B.M.) fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.



(b) Simulate new networks from latent variable model fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.



(c) Simulate new networks from latent variable model fit to check model goodness of fit on parameter edge density. Number of simulations = 20. Dimensions of latent space = 2.

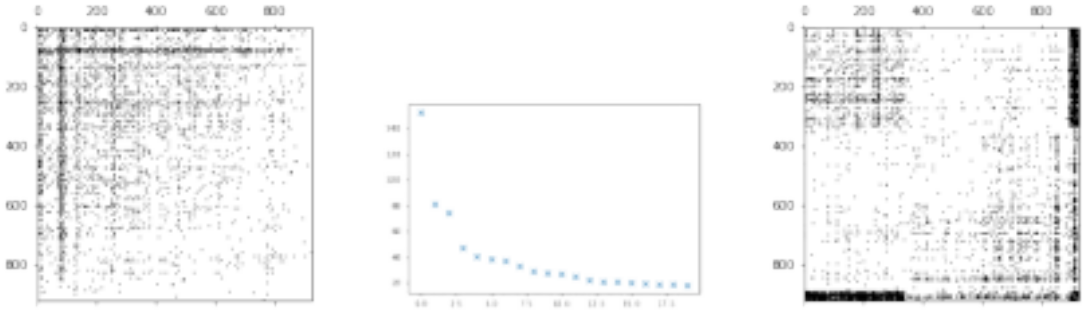
Figure 3.2: Fitting S.B.M. and latent variable model to Twt-Net data-set

3.3.2 Google+

Gplus-Net is a data-set of 923 Google+ users of a community and the "follower-following" relationships between them. The average clustering coefficient, i.e., transitivity between the members is 0.3. The high transitivity, coupled with a low average path length of 2.58, suggests the possibility of rapid information diffusion between members. Users of Google+ prefer following popular users; therefore, the network has negative assortativity and high inequality of degree (Gini index = 0.52). Other

characteristics observed in Gplus-Net are a large number of social contacts (average degree = 85). This leads to low average path length and diameter and high edge density. Edge density is lower compared to Twt-Net, which could imply that users are less active on Google+ than Twitter.

Applying the latent variable model to obtain the L.S.R. of Gplus-Net is infeasible due to the scale of the network. Therefore, only S.B.M. was fit to the data to estimate the block probabilities following the procedure outlined in Algorithm 1.



(a) Adjacency matrix of Google+ website data-set Gplus-Net

(b) Plot of singular values of Google+ website data-set Gplus-Net

(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.3: Analysis of of Google+ website data-set Gplus-Net using S.B.M.

Using eigen-gap heuristic, three latent classes are observed in Figure 3.3b. Figure 3.3c shows the re-ordered adjacency matrix with three latent classes. Using the edge probabilities at the block level, new networks are simulated to check model goodness of fit.

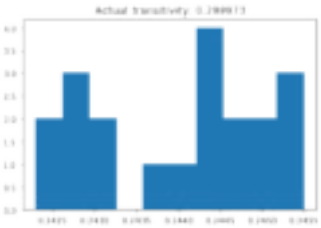


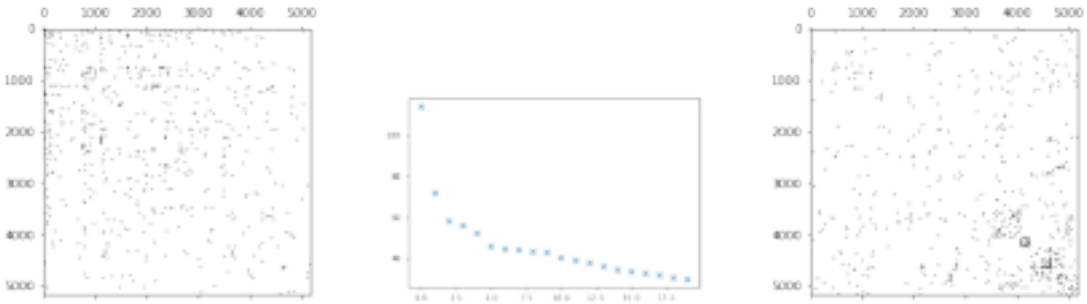
Figure 3.4: Simulate new networks from S.B.M. fit to check model goodness of fit on Gplus-Net for parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

After the block probabilities are estimated, the goodness of fit was checked. Figure

3.4 shows the range of transitivity of the simulated networks is 0.242-0.245 (upper and lower bounds of the 95% confidence interval). The actual transitivity of the network is 0.3, which is 0.05 more than the range of transitivity of the simulated networks. S.B.M. has generated L.S.R. that regenerate the original network (Gplus-Net) effectively. The posterior predictive check reveals that L.S.R. adequately captures the transitivity of the original network.

3.3.3 Blog

Blog-Net is a data-set of 5196 users and the relationships captured in the networks are of users commenting on blogs by other users. The average clustering coefficient, i.e., transitivity of the network is 0.08. S.B.M. is preferred for models with low transitivity. The latent variable model will not be able to scale to Blog-Net as the nodes are in the order of $\sim 10^3$. The network has node attributes, but S.B.M. does not model attributes. The adjacency matrix, as shown in Figure 3.5a, does not reveal the apparent possibilities of communities in the network.



(a) Adjacency matrix of Blog.com website data-set Blog-Net

(b) Plot of singular values of Blog.com website data-set Blog-Net

(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.5: Analysis of Blog.com website data-set Blog-Net using S.B.M.

Using eigen-gap heuristic, five latent classes are observed in Figure 3.5b. Figure 3.5c shows the re-ordered adjacency matrix with five latent classes. Using the edge probabilities at the block level, new networks are simulated to check model goodness of fit.

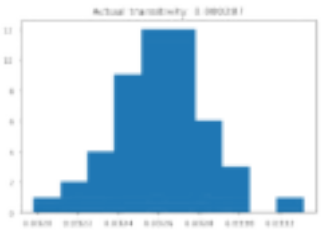


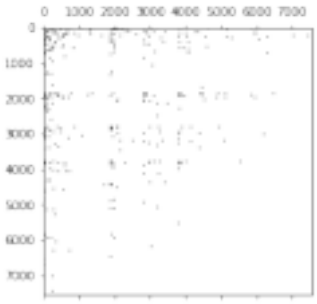
Figure 3.6: Simulate new networks from S.B.M. fit to check model goodness of fit on Blog-Net for parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

Figure 3.6 shows that S.B.M. has generated networks that have transitivity in the range of 0.032-0.033 (upper and lower bounds of the 95% confidence interval), whereas the actual transitivity of the original network (0.08) is 0.05 higher. Thus, S.B.M. regenerates the original network (Blog-Net) effectively. The posterior predictive check reveals that L.S.R. is adequately capturing the transitivity of the original network. However, the disadvantage of S.B.M. is that node attributes in the original network were not considered by the model. Thus, S.B.M. is not suitable for modeling networks with attribute information.

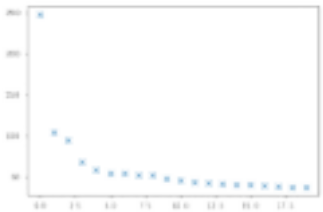
3.3.4 Flickr

Flickr-Net is a data-set of 7575 users and the relationships captured in the networks are of users "following" the profiles of other users. The average clustering coefficient, i.e., transitivity of the network is 0.1. Similar to online social networking websites like Twitter and Google+, Flickr-Net also has high Gini-index (0.67), negative assortativity (-0.23), high average degree (63.3), low diameter (2) and low average path length (2.15).

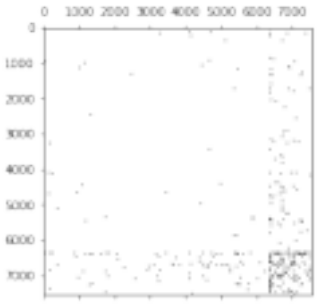
The latent variable model is not feasible for Flickr-Net due to a large number of nodes, so only S.B.M. was fit to the data for analysis. The adjacency matrix, as shown in Figure 3.7a, reveals several dense regions in the network.



(a) Adjacency matrix of Flickr.com website data-set Flickr-Net



(b) Plot of singular values of Flickr.com website data-set Flickr-Net



(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.7: Analysis of Flickr.com website data-set Flickr-Net using S.B.M.

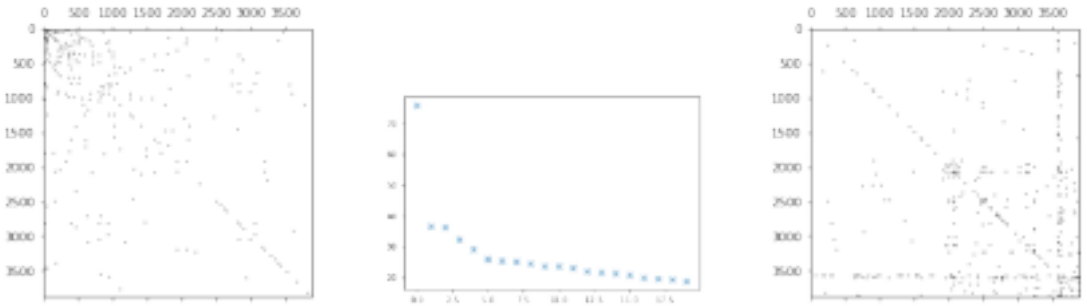
Using eigen-gap heuristic, three latent classes are observed in Figure 3.7b. Figure 3.7c shows the re-ordered adjacency matrix with five latent classes. Using the edge probabilities at the block level, new networks are simulated to check model goodness of fit.

Figure 3.8: Simulate new networks from S.B.M. fit to check model goodness of fit on Flickr-Net for parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

Figure 3.8 shows that the range of transitivity of the regenerated networks obtained from the simulation is 0.068-0.071 (upper and lower bounds of the 95% confidence interval), whereas the actual transitivity of the original network is 0.1 (0.03 higher). Thus, S.B.M. has generated L.S.R. that regenerate the original network (Flickr-Net) effectively. The posterior predictive check reveals that L.S.R. is adequately capturing the transitivity of the original network.

3.3.5 Protein protein interaction

Protein-Net is a data-set of 3890 proteins and their interactions with each other. The average clustering coefficient, i.e., transitivity of the network is 0.09. On average, a protein interacts with up to 20 other proteins. However, high Gini-index indicates that the majority of the interactions are concentrated amongst a section of proteins ($\sim$ 63%). Fitting latent variable model is not feasible for Protein-Net due to a large number of nodes, and hence only S.B.M. was fit to the data for analysis. The adjacency matrix, as shown in Figure 3.9a, reveals a dense cluster of nodes in the network.



(a) Adjacency matrix of Protein-protein interaction Protein-Net

(b) Plot of singular values of Protein-protein interaction Protein-Net

(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.9: Analysis of Protein-protein interaction Protein-Net data-set using S.B.M.

Using eigen-gap heuristic, two latent classes are observed in Figure 3.9b. Figure 3.9c shows the re-ordered adjacency matrix with two latent classes. Using the edge probabilities at the block level, new networks are simulated to check model goodness of fit.

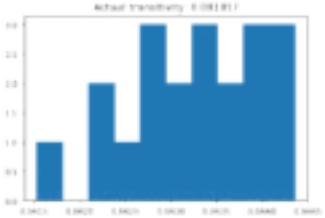
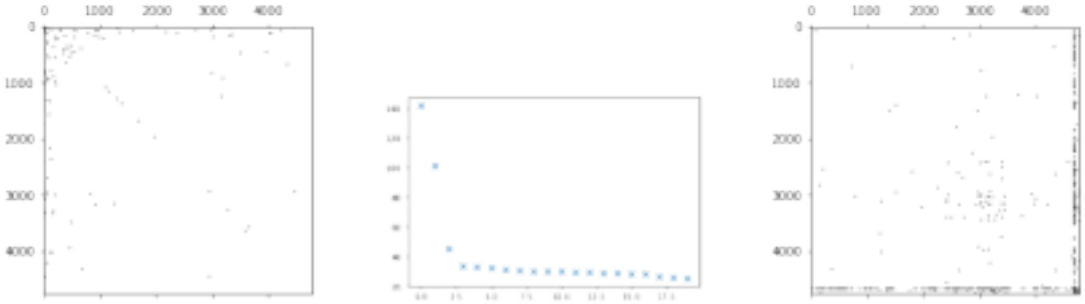


Figure 3.10: Simulate new networks from S.B.M. fit to check model goodness of fit on Protein-Net for parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

Figure 3.10 gives the simulation results that reveal the range of transitivity of the simulated networks as 0.04-0.044 (upper and lower bounds of the 95% confidence interval), which is 0.05 lower than the actual transitivity of the original network. Thus, S.B.M. has generated L.S.R. that regenerate the original network (Protein-Net) effectively. The posterior predictive check reveals that L.S.R. is adequately capturing the transitivity of the original network.

3.3.6 Wikipedia

Wiki-Net is a data-set of 4777 and their hyperlinks to other web-pages in the network. The average clustering coefficient, i.e., transitivity of the network is 0.43. The adjacency matrix, as shown in Figure 3.11a, reveals a dense cluster of nodes in the network.



(a) Adjacency matrix of Wikipedia.com data-set Wiki-Net

(b) Plot of singular values of Wikipedia.com data-set Wiki-Net

(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.11: Analysis of of Wikipedia.com data-set Wiki-Net using S.B.M.

Using eigen-gap heuristic, three latent classes are observed in Figure 3.11b. Figure 3.11c shows the re-ordered adjacency matrix with three latent classes. Using the edge probabilities at the block level, new networks are simulated to check model goodness of fit.

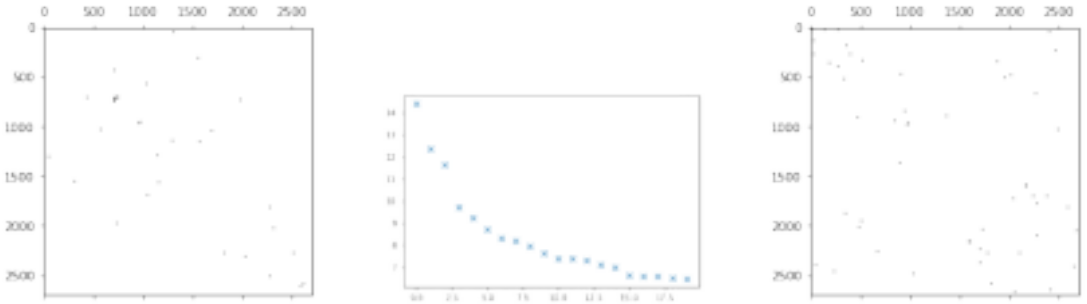
Figure 3.12: Simulate new networks from S.B.M. fit to check model goodness of fit on Wiki-Net for parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

Figure 3.12 shows that regenerated networks from the simulations have transitivity in the range of 0.34-0.39 (upper and lower bounds of the 95% confidence interval), whereas the actual transitivity of the original network is 0.453 (0.08-0.12 higher). Thus, S.B.M. has generated L.S.R. that does not regenerate the original network (Wiki-Net) effectively. The posterior predictive check reveals that L.S.R. is not adequately capturing the transitivity of the original network.

3.3.7 Cora-Net

Cora-Net is a data-set of 2708 academic papers and the citations between them. The average clustering coefficient, i.e., transitivity of the network is 0.09. Transitivity is lower for citation networks as compared to social networks as i citing j and j citing k might not lead to k citing i.

S.B.M. is preferred for models with low transitivity. The latent variable model will not be able to scale to Cora-Net as the nodes are in the order of $\sim 10^3$. The network is sparse, i.e., $|V| \sim |E|$ and the adjacency matrix, as shown in Figure 3.13a does not reveal apparent possibilities of communities in the network.



(a) Adjacency matrix of CORA Research Paper Classification Data-set Cora-Net

(b) Plot of singular values of CORA Research Paper Classification Data-set Cora-Net

(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.13: Analysis of CORA Research Paper Classification Data-set Cora-Net using S.B.M.

Using eigen-gap heuristic, three latent classes are observed in Figure 3.13b. Figure 3.13c shows the re-ordered adjacency matrix with three latent classes. Using the edge probabilities at the block level, new networks are simulated to check model goodness of fit.

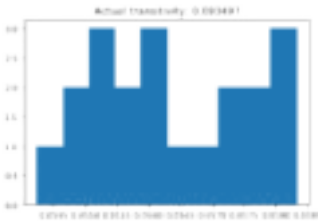


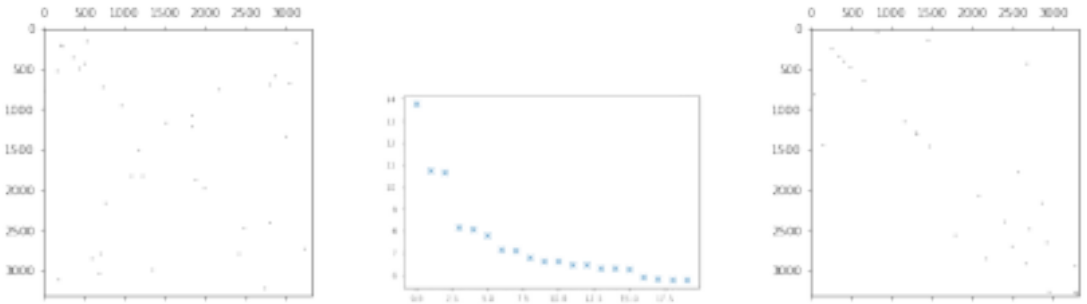
Figure 3.14: Simulate new networks from S.B.M. fit to check model goodness of fit on Cora-Net for parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

Figure 3.14 shows that the range of transitivity of the simulated networks as 0.0145-0.0185 (upper and lower bounds of the 95% confidence interval), whereas the actual transitivity of the original network is 0.093 (higher by 0.08). Thus, S.B.M. has generated L.S.R. that does not regenerate the original network (Cora-Net) effectively. The posterior predictive check reveals that L.S.R. is not adequately capturing the transitivity of the original network.

3.3.8 CiteSeer

Cite-Net is a data-set of 3312 academic papers and the citations between them. The average clustering coefficient, i.e., transitivity of the network is 0.13. Transitivity is lower for citation networks as i citing j and j citing k might not lead to k citing i.

S.B.M. is preferred for models with low transitivity. The latent variable model will not be able to scale to Cite-Net as the nodes are in the order of $\sim 10^3$. The network is sparse, i.e., $|V| \sim |E|$ and the adjacency matrix, as shown in Figure 3.15a does not reveal apparent possibilities of communities in the network.



(a) Adjacency matrix of Citeseer.com data-set Cite-Net

(b) Plot of singular values of Citeseer.com data-set Cite-Net

(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.15: Analysis of Citeseer.com data-set Cite-Net using S.B.M.

Using eigen-gap heuristic, five latent classes are observed in Figure 3.15b. Figure 3.15c shows the re-ordered adjacency matrix with five latent classes. Using the edge probabilities at the block level, new networks are simulated to check model goodness of fit.

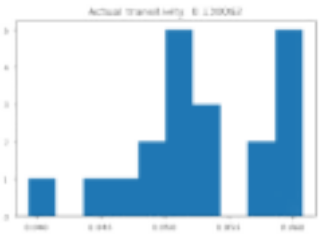
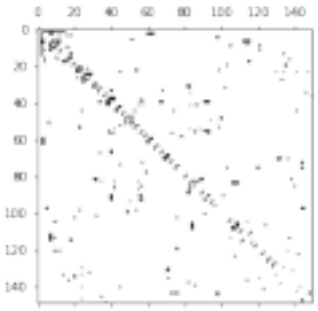


Figure 3.16: Simulate new networks from S.B.M. fit to check model goodness of fit on Cite-Net for parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

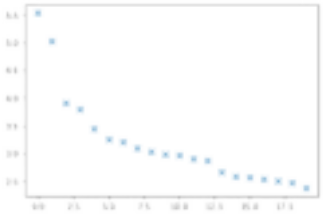
Figure 3.16 shows the range of transitivities of the simulated models as 0.04-0.06 (upper and lower bounds of the 95% confidence interval), which is 0.07 less than the actual transitivity of the original network. Thus, S.B.M. has generated L.S.R. that does not regenerate the original network (Cite-Net) effectively. The posterior predictive check reveals that L.S.R. is not adequately capturing the transitivity of the original network.

3.3.9 Highway

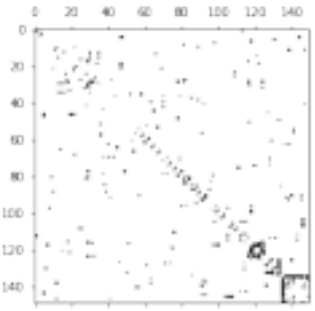
High-Net is a data-set of 205 cities and the highways that connect them to other cities in the network. The average clustering coefficient, i.e., transitivity between the members is 0.28. Figure 3.18 shows the results of fitting S.B.M. to High-Net using the procedure outlined in Algorithm 1. Figure 3.17a shows presence of multiple dense regions (latent classes) in the plot of adjacency matrix A. Hence, to choose the latent classes, we examine the singular values of A.



(a) Adjacency matrix of Transportation network data-set High-Net



(b) Plot of singular values of Transportation network data-set High-Net



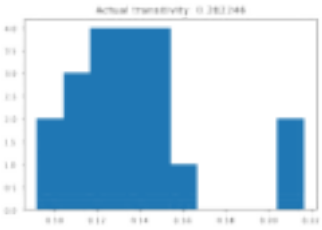
(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.17: Analysis of Transportation network data-set High-Net using S.B.M.

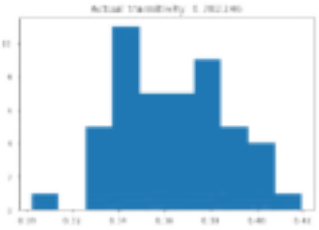
Using eigen-gap heuristic, five latent classes are observed in Figure 3.17b. The nodes in these latent classes are assigned class-memberships, and then the adjacency matrix is re-ordered. Figure 3.17c shows the re-ordered adjacency matrix with five latent classes. Once the class-memberships are assigned, the edge probabilities at the block level are calculated. Finally, new networks are simulated from edge probabilities to check model goodness of fit.

Figure 3.18a shows the range of transitivity of the simulated networks from S.B.M. as 0.09-0.16 (upper and lower bounds of the 95% confidence interval), which is 0.12 less than the actual transitivity of the original network. Thus, S.B.M. has generated L.S.R. that does not regenerate the original network (High-Net) effectively. The L.S.R. is not adequately capturing the transitivity of the original networks. The range of transitivities of the simulated networks obtained from the latent variable model is 0.3-0.42 (upper and lower bounds of the 95% confidence interval), which is 0.02 higher than the actual transitivity of the original network, as shown in Figure 3.18b. Similarly, the range of densities of the simulated networks is 0.016-0.019 (upper and lower bounds of the 95% confidence interval), as shown in Figure 3.18c. This is close to the actual edge density of the original network, which is 0.018. The latent variable model was able to generate L.S.R. that could replicate the transitivity (as shown in Figure 3.18b) and edge density (as shown in Figure 3.18c) of the original network better than S.B.M.s. Conclusions from the posterior predictive check of S.B.M. and latent variable model is that L.S.R.

generated using latent variable model are more effective than S.B.M. in High-Net.



(a) Simulate new networks from data-set High-Net - S.B.M. fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.



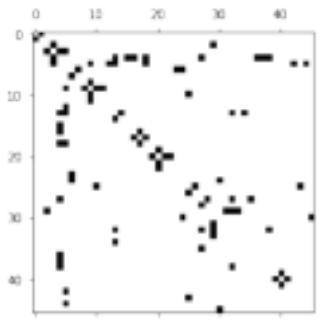
(b) Simulate new networks from Latent variable model fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

(c) Simulate new networks from Latent variable model fit to check model goodness of fit on parameter edge density. Number of simulations = 20. Dimensions of latent space = 2.

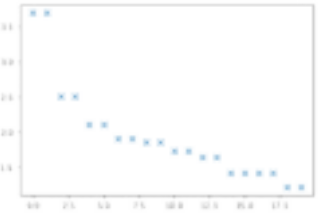
Figure 3.18: Fitting S.B.M. and Latent variable model to High-Net data-set

3.3.10 Grey's anatomy

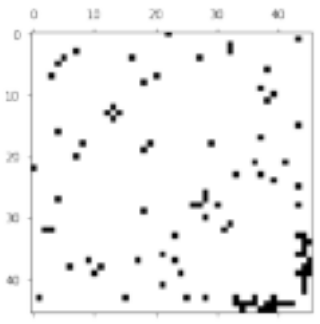
Grey-Net is a data-set of 44 actors in the popular series Grey's Anatomy and the "sexual" relationships between them. Figure 3.20 shows the results of fitting S.B.M. to Grey-Net using the procedure outlined in Algorithm 1. Figure 3.19a shows a dense adjacent matrix A but no presence of communities (latent classes) can be detected in the plot. Hence, to choose the latent classes, we examine the singular values of A.



(a) Adjacency matrix of Grey's anatomy data-set Grey-Net



(b) Plot of singular values of Grey's anatomy data-set Grey-Net

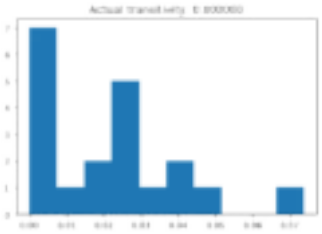


(c) Re-order nodes in adjacency matrix by class memberships

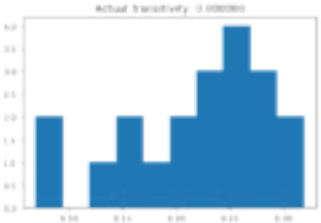
Figure 3.19: Analysis of Grey's anatomy data-set Grey-Net using S.B.M.

Using eigen-gap heuristic, seven latent classes are observed in Figure 3.19b. The nodes in these latent classes are assigned class-memberships, and then the adjacency matrix is re-ordered. Figure 3.19c shows the re-ordered adjacency matrix with seven latent classes. Once the class-memberships are assigned, the edge probabilities at the block level are calculated. Finally, new networks are simulated from edge probabilities to check model goodness of fit.

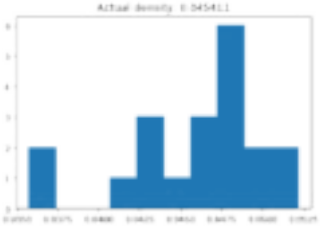
Figure 3.20a shows the range of transitivities of the simulated networks from S.B.M. to 0.0-0.05 (upper and lower bounds of the 95% confidence interval), which is 0.05 higher than the actual transitivity of the original network. Thus, S.B.M. has generated L.S.R. that regenerate the original network (Grey-Net) effectively. The L.S.R. can effectively capture the transitivity of the original network. Figure 3.20b shows the range of transitivities of the simulated networks from L.V.M. as 0.125-0.35 (upper and lower bounds of the 95% confidence interval), which is, on average, 0.35 higher than the actual transitivity of the original network. Figure 3.20c shows the range of densities of the simulated networks from the latent variable model to 0.04-0.05 (upper and lower bounds of the 95% confidence interval), which is close to the actual edge density (0.045) of the original network. The latent variable model was not able to generate L.S.R. that could replicate the transitivity (as shown in Figure 3.20b) even though the fit to the edge density (as shown in Figure 3.20c) of the original network was correct. Conclusions from the posterior predictive check of S.B.M. and L.V.M. models is that L.S.R. generated using S.B.M. are more effective than L.V.M. in Grey-Net.



(a) Simulate new networks of Grey-Net data-set from S.B.M. fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.



(b) Simulate new networks from Latent variable model fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

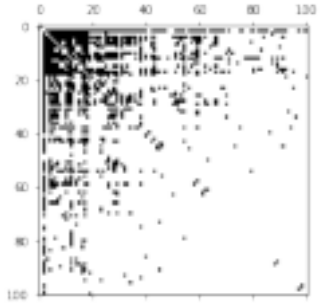


(c) Simulate new networks from Latent variable model fit to check model goodness of fit on parameter edge density. Number of simulations = 20. Dimensions of latent space = 2.

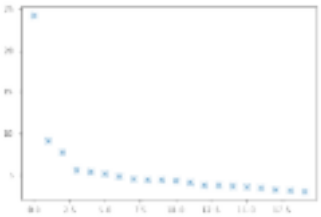
Figure 3.20: Fitting S.B.M. and Latent variable model to Grey-Net data-set

3.3.11 Trade

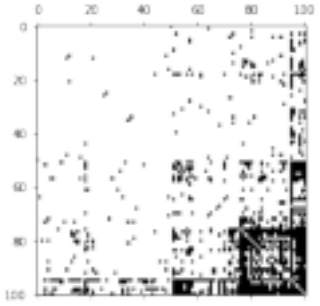
Trade-Net is a data-set of 99 countries and their trading ties. The average clustering coefficient, i.e., transitivity between the members is 0.44. Figure 3.22 shows the results of fitting S.B.M. to Trade-Net using the procedure outlined in Algorithm 1. Figure 3.21a shows a dense adjacent matrix A. Hence, to choose the latent classes, a plot of the singular values of A is created.



(a) Adjacency matrix of Trade network data-set Trade-Net



(b) Plot of singular values of Trade network data-set Trade-Net



(c) Re-order nodes in adjacency matrix by class memberships

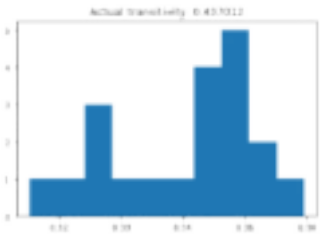
Figure 3.21: Analysis of Trade network data-set Trade-Net using S.B.M.

Using eigen-gap heuristic, three latent classes are observed in Figure 3.21b. The nodes

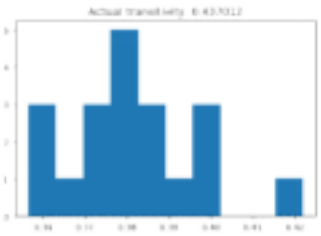
in these latent classes are assigned class-memberships, and then the adjacency matrix is re-ordered. Figure 3.21c shows the re-ordered adjacency matrix with three latent classes. Once the class-memberships are assigned, the edge probabilities at the block level are calculated. Finally, new networks are simulated from edge probabilities to check model goodness of fit.

Figure 3.22a shows the range of transitivities of the simulated networks from S.B.M. as 0.31-0.36 (upper and lower bounds of the 95% confidence interval), which is 0.07 lower than the actual transitivity of the original network (0.43). Thus, S.B.M. has generated L.S.R. that does not regenerate the original network (Trade-Net) effectively. The L.S.R. is not adequately capturing the transitivity of the original networks. Figure 3.22b shows the range of transitivities of the simulated networks from the latent variable model to 0.36-0.42 (upper and lower bounds of the 95% confidence interval), which is 0.04 lower than the actual transitivity of the original network (0.43). Figure 3.22c shows the range of densities of the simulated networks from the latent variable model as 0.108-0.118 (upper and lower bounds of the 95% confidence interval), which contains the actual edge density of the original network (0.113).

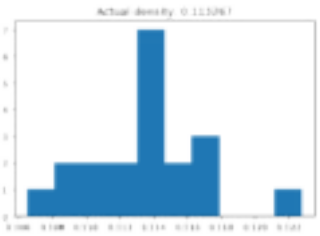
Latent variable model was able to generate L.S.R. that could replicate the transitivity (as shown in Figure 3.22b) and edge density (as shown in Figure 3.22c) of the original network better than S.B.M. Conclusions from the posterior predictive check of S.B.M. and Latent variable model models is that L.S.R. generated using Latent variable model are more effective than S.B.M. in Trade-Net.



(a) Simulate new networks of Trade-Net data-set from S.B.M. fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.



(b) Simulate new networks from Latent variable model fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

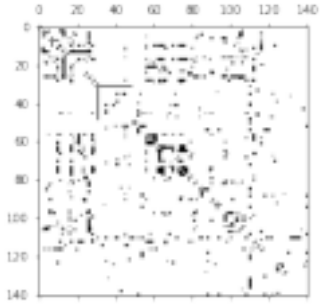


(c) Simulate new networks from Latent variable model fit to check model goodness of fit on parameter edge density. Number of simulations = 20. Dimensions of latent space = 2.

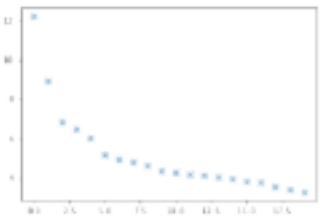
Figure 3.22: Fitting S.B.M. and Latent variable model to Trade-Net data-set

3.3.12 Bill co-sponsorship

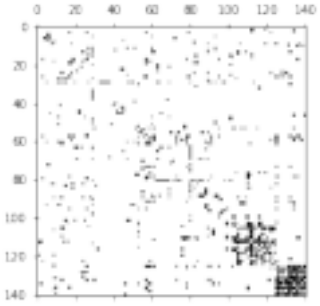
Bill-Net is a data-set of 139 legislators that have co-sponsored legislation with each other. The average clustering coefficient, i.e., transitivity between the members is 0.32. Figure 3.2 shows the results of fitting S.B.M. to Bill-Net using the procedure outlined in Algorithm 1. Figure 3.23a shows a dense adjacent matrix A but to choose the latent classes, the plot of singular values of A is needed.



(a) Adjacency matrix of Bill co-sponsorship data-set Bill-Net



(b) Plot of singular values of Bill co-sponsorship data-set Bill-Net



(c) Re-order nodes in adjacency matrix by class memberships

Figure 3.23: Analysis of Bill co-sponsorship data-set Bill-Net using S.B.M.

Using eigen-gap heuristic, four latent classes are observed in Figure 3.23b. The nodes

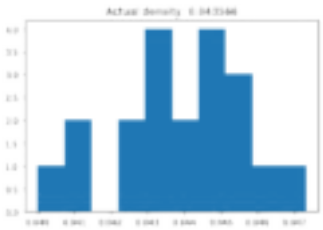
in these latent classes are assigned class-memberships, and then the adjacency matrix is re-ordered. Figure 3.23c shows the re-ordered adjacency matrix with four latent classes. Once the class-memberships are assigned, the edge probabilities at the block level are calculated. Finally, new networks are simulated from edge probabilities to check model goodness of fit.

Figure 3.24a shows the range of transitivities of the simulated networks from S.B.M. as 0.19-0.28 (upper and lower bounds of the 95% confidence interval), which is 0.10 lower than the actual transitivity of the original network (0.32). Figure 3.24b shows the range of transitivities of the simulated networks from the Latent variable model as 0.26-0.33 (upper and lower bounds of the 95% confidence interval), which is captures the actual transitivity of the original network (0.32). Figure 3.24c shows the range of densities of the simulated networks from Latent variable model as 0.04-0.047 (upper and lower bounds of the 95% confidence interval), which captures the actual edge density of the original network (0.043).

Thus, S.B.M. has generated L.S.R. that does not regenerate the original network (Bill-Net) effectively as a latent variable model. The L.S.R. is not adequately capturing the transitivity of the original networks as a latent variable model. Latent variable model was able to generate L.S.R. that could replicate the transitivity (as shown in Figure 3.24b) and edge density (as shown in Figure 3.24c) of the original network better than S.B.M. Conclusions from the posterior predictive check of S.B.M. and Latent variable model is that L.S.R. generated using Latent variable model are more effective than S.B.M. in Bill-Net.

(a) Simulate new networks of Bill-Net data-set from S.B.M. fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.

(b) Simulate new networks from Latent variable model fit to check model goodness of fit on parameter transitivity. Number of simulations = 20. Dimensions of latent space = 2.



(c) Simulate new networks from Latent variable model fit to check model goodness of fit on parameter edge density. Number of simulations = 20. Dimensions of latent space = 2.

Figure 3.24: Fitting S.B.M. and Latent variable model to Bill-Net data-set

3.4 Discussion of Results and Summary

Network analysis is a crucial aspect of computational social science. The omnipresent nature of graphs (networks) in the world has further enhanced the importance of this field. Although networks are present in every domain, their analysis revealed that the structural characteristics shared by them are similar, i.e., low average path, low diameter, etc. It is further revealed that networks also capture the behavior of the entities present in them. Using concepts of graph theory, it is possible to make a statistically valid analysis of these systems and provide insights into their growth.

Networks across different domains saw low edge density and the presence of inequality. Networks such as Twt-Net, Gplus-Net, Flickr-Net, Wiki-Net, Blog-Net, Grey-Net and Bill-Net are a particular type of network called "social networks". Social networks represent the sum of all professional, friendship, or family ties of the actors involved in them. Social networks were observed to have higher edge density and average degree compared to other networks. They also had high transitivity, low diameter, and negative assortativity.

Statistical models such as Stochastic Block Model (S.B.M.) and Latent variable model

were fit to various application scenarios. Table 3.4 provides a summary of the results of the Stochastic Block Model (S.B.M.) and Latent variable model on data-sets. It provides the actual transitivity of the networks (Actual c) and the mean transitivities of the simulated networks from S.B.M. (Mean c_1) and latent variable model (Mean c_2). The values of Mean c_1 and Mean c_2 along with limits of two standard deviations $2SD$ of the mean (upper and lower bounds of the 95% confidence interval) are used to understand which model was comparatively more effective i.e., within ±0.05, in fitting the data. In Table 3.4, '*' is used to indicate that Latent variable model was infeasible for the dataset. '-' is used to indicate that neither S.B.M. nor latent variable model could produce an effective fit on data.

Table 3.4: Summary of results of Stochastic Block Model (S.B.M.) and latent variable model on data-sets

Data-set	**Actual** c	**Mean** c_1 ($\pm 2SD$)	**Mean** c_2 ($\pm 2SD$)	**Effective fit** (±0.05)
Twt-Net	0.44	0.31 (0.3 - 0.33)	0.4 (0.4-0.404)	Latent variable model
Gplus-Net	0.3	0.243 (0.242-0.245)	*	S.B.M.
Blog-Net	0.08	0.03 (0.032-0.033)	*	S.B.M.
Flickr-Net	0.1	0.07 (0.068-0.071)	*	S.B.M.
Protein-Net	0.09	0.04 (0.04-0.04)	*	S.B.M.
Wiki-Net	0.43	0.37 (0.34-0.39)	*	-
Cora-Net	0.09	0.01 (0.0145-0.0185)	*	-
Cite-Net	0.13	0.05 (0.04-0.06)	*	-
High-Net	0.28	0.13 (0.09-0.16)	0.32 (0.3-0.42)	Latent variable model
Grey-Net	0	0.02 (0.0-0.05)	0.2 (0.125-0.35)	S.B.M.
Trade-Net	0.43	0.33 (0.31-0.36)	0.38 (0.36-0.42)	Latent variable model
Bill-Net	0.32	0.24 (0.19-0.28)	0.31 (0.26-0.33)	Latent variable model

For data-sets such as Twt-Net, High-Net, Trade-Net and Bill-Net where transitivity is >0.3, latent variable model fit better than S.B.M. as it is more suitable to model assortative mixing i.e., transitivity. Whereas, for Gplus-Net, Blog-Net, Flickr-Net, Protein-Net and Grey-Net, S.B.M. was more effective as these networks had low transitivity <0.3. However, both models ignore the attributes associated with the networks. Hence, the results on networks with attributes were mixed. None of the models were observed to regenerate networks with transitivity in the acceptable limits of ±0.05 for Wiki-Net, Cora-Net and Cite-Net. The computational complexity of S.B.M. is $\Theta(|V| * |E| * d)$

and latent variable model is $\Theta(|V| * |V| * d)$. Thus, S.B.M. is applicable for networks with nodes in a range of 10^3, whereas the Latent variable model is feasible for networks with a few hundred nodes. Hence, it is necessary to investigate models that can scale to larger networks (10^3+).

Chapter 4

ENSEMBLE MODEL FOR NETWORK REPRESENTATION LEARNING

4.1 Introduction

Network science has been applied to understand the trends in domains of scientific literature such as social networking websites (Twitter, Google+), blogs (Blog.com), photo-sharing websites (Flickr), protein-protein interactions (Protein-Net), citation networks (CORA and CiteSeer), transportation networks (High-Net), sexual contact network (Grey-Net), trade network (Trade-Net) and bill co-sponsorship network (Bill-Net). Trends identified using network analysis are the rate of information diffusion, the dominance of certain entities, the number of social contacts and community structure. These trends would not be easily identifiable using relational databases. However, calculating graph statistics such as Gini index, average path length, diameter etc. is computationally intensive. Earlier, network analysis required graph statistics to build features for training the learning model. However, as graphs increased in size this feature engineering step created bottlenecks. Hence we use latent space representation of such networks. With L.S.R., it is possible to develop downstream machine learning applications such as node clustering, node classification, expert prediction, etc without feature engineering. Therefore, there has been a paradigm shift in the machine learning

community towards representation learning on graphs instead of feature engineering [119].

The application of statistical models for latent space representation revealed certain drawbacks such as limited applicability, computation cost, reliance upon expensive and often unstable methods for probabilistic inference [69, 72]. To overcome these drawbacks, latent space representation techniques were proposed. Chapter 2 discussed the taxonomy of L.S.R. techniques present in literature. Three broad categories of L.S.R. techniques for networks exist: Probabilistic models, Statistical models, and Network representation learning models. The limited flexibility of probabilistic models resulted in poor fits to data. Even though probabilistic models were "generative," they did not generate networks that shared many properties with the specific network they were fit to. Statistical models such as S.B.M. and latent variable model were feasible for data-sets with nodes in the range of 10^{1-3}. This was due to the computationally costly stochastic calculations in such models. N.R.L. methods have an encoder function that learns a “similarity" between the nodes of the network representation of a system. This “similarity" is then preserved by the encoder in the node embeddings (vector representations) that are generated by it. The focus of this chapter is to improve the latent space representations of networks with the use of enhancements in N.R.L. methods. N.R.L. methods are suitable for network analysis as the networks in this chapter are large, with nodes in the range of 10^3.

Section 4.2 provides a background of N.R.L techniques and states their drawbacks. The motivation for the proposed ensemble approach is provided in Section 4.2.1. Section 4.3 describes the mathematical model followed by experimental study in Section 4.4 and the chapter is concluded in Section 4.5.

4.2 Background

N.R.L. frameworks consist of an encoder that learns to preserve a similarity measure between nodes in a network. This similarity measure can be adjacency in the network, neighborhood overlap between the nodes, reachability within multiple hops or

co-occurrence on random walks. Several authors have proposed different versions of similarity, and this has consequentially led to a proliferation of N.R.L. frameworks in the literature. A majority of these frameworks can be categorized into three groups based on the similarity measure that they capture in their latent representations. These classes are Adjacency preserving methods [24, 25, 26, 30, 31, 84, 85, 86, 87, 88], Multi-hop distance preserving methods and Random walk occurrence preserving methods [89, 90, 91, 92, 93, 94, 95, 96, 97, 98].

Adjacency preserving methods define the similarity between two nodes u, v in the original network as the presence of an edge between them in the adjacency matrix A, i.e., $A_{u,v} = 1$. Multi-hop distance preserving techniques define similarity in the original network as the existence of a k-hop path between two nodes u, v, i.e., $A^k_{u,v} = 1$ where $A^k_{u,v}$ is the k-hop adjacency matrix. Neighborhood overlap preserving techniques define similarity in the original network as the degree of overlap of neighborhoods N of two nodes $N_u \cap N_v$. Random walk occurrence preserving techniques define similarity in the original network as the probability of reaching a node v by a random walk from node u, i.e., $P(v|u)$.

The drawback of adjacency-preserving techniques is that they generate node embedding that over-fit the adjacency matrix of the original graph. This causes failure in detecting inconspicuous connections. Deterministic node similarity measures have to be hand-designed, which is the drawback of neighborhood overlap preserving techniques and multi-hop distance preserving techniques. The performance of random walk co-occurrence preserving techniques depends on the strategy used for random walk viz. plain random walks, skipping, D.F.S., B.F.S., etc.

The frameworks in the literature preserve one of the many proximity measures defined on the graph. This could lead to node embedding that is favorable for specific applications but unsuitable for others [48, 49, 50]. There is strong evidence that a single model can be outperformed by an ensemble of models [120]. Two primary approaches for ensembling models are Combine by learning and Combine by consensus [120]. Combine by consensus approach is suitable for unsupervised learning tasks such as L.S.R. In combine by consensus technique, the representations of the base models are

unified by consensus to obtain the final representations. The three base models are MF (adjacency preserving methods), LINE (multi-hop distance preserving) and DeepWalk (random walk occurrence preserving) methods. The selection of base models is based on their time complexity to ensure scalability to network data-sets with with nodes beyond 10^3.

4.2.1 Motivation

The current inquiry experiments with an ensemble model for generating effective L.S.R. The aim is to avoid the shortcoming of a single model by relying on a combination of multiple frameworks. An extensive survey of N.R.L. techniques did not reveal the use of ensemble models for representation learning. As a single model is unlikely to capture the entire underlying structure of the data to achieve optimal representations, integrating multiple models may improve the accuracy of representation learning significantly. This gap serves as the principal motivation for this study. The focus in the current inquiry is on an ensemble model in which representation learning of base models are combined by consensus using a weighted average.

4.2.2 Summary of contributions

In this chapter, experiments are performed using a weighted average based ensemble model for latent space representation of networks. The N.R.L. methods belonging to three different categories are utilized in the ensemble model.

4.3 Ensemble Network Representation Learning Framework

Three separate network representation learning frameworks are used to obtain the initial node level embeddings.

4.3.1 Calculate node embeddings through adjacency preserving similarity (MF)

The adjacency matrix A_{m*n} is factored into low rank representations B_1, B_2 such that $A \sim B_1 B_2$ [31]. A non-negative matrix factorization approach (Eq. 4.1) ensures topological information is captured in vector embedding.

$$B_1, B_2 = ||A - B_1 B_2||_F^2 + \alpha_1 ||B_1||_F^2 + \alpha_2 ||B_2||_F^2 \tag{4.1}$$

where,

1. $||.||_F^2$ - Frobenius norm
2. $\alpha_1, \alpha_2 \geq 0$ - Weight parameters

4.3.2 Calculate node embeddings through multi-hop distance preserving similarity (LINE)

The objective function is to obtain node embeddings that capture second-order proximity. The L2-norm of the loss function, which needs to be minimized is given in Eq. 4.2.

$$min||S - U^s.U^{tT}||_F^2 \tag{4.2}$$

$$S = M_g^{-1}.M_t \tag{4.3}$$

$$M_g = I - \beta.A \tag{4.4}$$

$$M_t = \beta.A \tag{4.5}$$

where,

1. S = high order proximity matrix in Eq. 4.2, Eq. 4.3
2. U^s, U^t = source and target embedding vectors
3. M_g, M_t = polynomial of adjacency matrix A in Eq. 4.4, Eq. 4.5
4. I = identity matrix, β = decay parameter of Katz index

4.3.3 Calculate node embeddings through random walk occurance preserving similarity (DeepWalk)

Random walks on the graph are performed and the results are provided to skip-gram neural network to obtain the node embeddings. The objective of the skip-gram model is as shown in Eq. 4.6:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c; j\neq 0} logp(w_{t+j}|w_t) \tag{4.6}$$

The basic skip-gram formulation defines $p(w_{t+j}|w_t)$ (probabilities of other vertices in the social network occurring within the context window of the current vertex) using the softmax function as shown in Eq. 4.7:

$$p(wout|win) = \frac{exp(v_{wout}^T * v_{win})}{\sum_{w=1}^{W} exp(v_{wout}^T * v_{win})} \tag{4.7}$$

where v_{win} and v_{wout} are the "input" and "output" vector representations of w and W is the number of words in the vocabulary.

4.3.4 Ensemble network representation learning framework

For obtaining a robust representation learning framework, an ensemble mechanism that takes the weighted mean of the individual models (MF, LINE and DeepWalk) is used. The final node level embeddings are obtained as given in Eq. 4.8. Z is the final L.S.R. of the network obtained by using combine by consensus technique, i.e., the representations of the base models (MF, LINE and DeepWalk) Z_i are unified by consensus to obtain the final representations. Algorithm 4 is a minimized algorithm of the proposed mechanism, and an abstract block diagram for the same is given in Fig. 4.1. Depending on the values of α, two consensus models are possible. If α is equal for all base models (MF, LINE and DeepWalk), then the ensemble model is a mean ensemble, and if α is different for each model, then it is a weighted-average ensemble.

$$Z = \sum_{i=1}^{N(Z)} \frac{\alpha_i Z_i}{|N(Z)|} \tag{4.8}$$

where,

1. $N(Z)$ = Number of base models.
2. α = weights of base models. Obtained for weighted-average ensemble through tuning.

Algorithm 4: Ensemble Network representation learning framework

Input: G = V, E

Output: Z = $R^{d*|V|}$

Calculate Cost function for adjacency preserving similarity as:
$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - A_{u,v}||^2$
Calculate Cost function for multi-hop distance preserving similarity as:
$L = \sum_{(u,v)\in(V*V)} ||z_u^T z_v - S_{u,v}||^2$
Calculate Cost function for random walk occurance preserving similarity as:
$L = \sum_{(u\in V)} \sum_{(v\in N_r(u))} -log(P(v|z_u))$
Optimize parameters of Z using stochastic gradient descent
Combine individual learned representations to final learned representation Z:
$\sum_{i=1}^{N(Z)} \frac{\alpha_i Z_i}{|N(Z)|}$

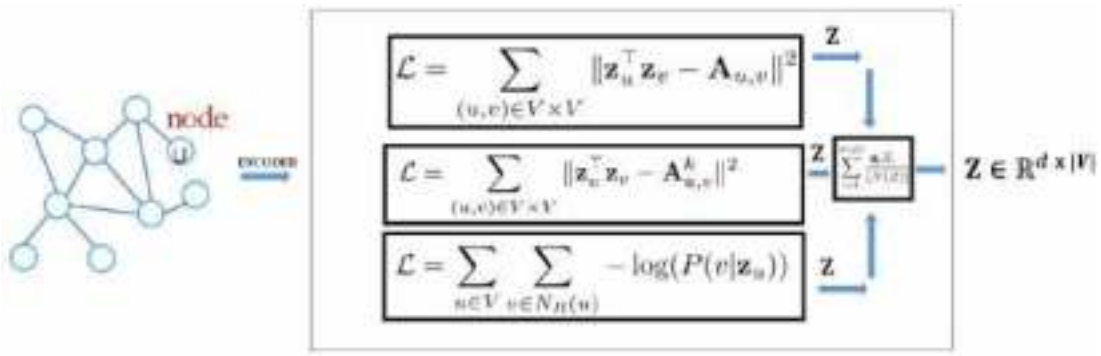


Figure 4.1: Ensemble network representation learning framework

4.4 Experimental Study

The proposed weighted-average and mean ensemble methods is evaluated on network data-sets to understand its performance vis-a-vis other N.R.L. techniques. The

weighted-average and mean ensemble methods convert network data of Blog-Net, Cora-Net, Cite-Net, Flickr-Net, Protein-Net and Wiki-Net from their original dimensions into vector space of dimension = 32. The quality of the vertex embeddings is measured using distance-based statistics obtained from clustering the representations. The performance metrics used are separation index, widest within-cluster gap, average silhouette width, the average distance between clusters and Dunn index. Twenty iterations are performed on vertex embeddings for calculating each performance metric. The values of mean along with limits of two standard deviations $2SD$ of the mean (upper and lower bounds of the 95% confidence interval) are given for quantitative comparison. Configuration of the system on which experimental study was conducted is Intel(R)Core(T.M.) i5-6402P CPU@2.8GHz with four cores and 8GB DDR3 RAM. ProNet-Core C++ framework was used for implementation network embedding techniques (MF, LINE and DeepWalk).

4.4.1 Performance metrics

4.4.1.1 Separation index

It is computed based on the distances for every point to the closest point not in the same cluster. The separation index is then the mean. A lower value indicates that clusters are well separated.

4.4.1.2 Widest within-cluster gap

Largest link in the within-cluster minimum spanning tree is computed. A more considerable value indicates useful clustering.

4.4.1.3 Average silhouette width

It is a measure of how similar an object is to its cluster compared to other clusters. The silhouette ranges from -1 to +1, where a high value indicates that the object is well matched to its cluster and poorly matched to neighboring clusters.

4.4.1.4 Average distance between clusters

For two clusters r, s average distance between clusters should be significant to indicate well-separated clusters.

$$L(r,s) = \frac{1}{n_r * n_s} \sum_{i=1}^{n_r} \sum_{i=1}^{n_s} D(x_{ri}, x_{sj}) \tag{4.9}$$

4.4.1.5 Dunn index

A lower Dunn index indicates better clustering. $\delta(C_i, C_j)$ is inter-cluster distance metric between clusters C_i and C_j, Δ_k is the maximum within cluster variance.

$$DI_m = \frac{\min_{1 \le i < j \le m} \delta(C_i, C_j)}{\max_{1 \le k \le m} \Delta_k} \tag{4.10}$$

4.4.2 Data-sets

Network data-sets with $|V| > 10^3$ were chosen for the evaluation of the ensemble technique.

Table 4.1: Description of Network Data-sets with binary attributes

Description	**Blog-Net**	**Flickr-Net**	**Protein-Net**	**Wiki-net**	**Cora-net**	**Cite-net**
$\|V\|$	5196	7575	3890	4777	2708	3312
V^a	8189	12047	50	40	1434	3704
$\|E\|$	171743	239738	37845	54810	5429	4732
c	0.08	0.1	0.09	0.43	0.14	0.13

4.4.3 Results

DeepWalk [89], LINE [24] and MF [27] are the baseline techniques used for comparison. Two ensemble methods: mean ensemble and weighted-average ensemble (WeightedAvg) are evaluated to understand the latent space representations obtained by them. Table 4.2 gives the weights of the individual models of the weighted-average ensemble. α_1, α_2 and α_3 are the weights of adjacency based similarity, multi-hop based similarity and random walk based similarity respectively.

Table 4.2: Weights of the base models of WeightedAvg

Data-set	**Mean** α_1 $(\pm 2SD)$	**Mean** α_2 $(\pm 2SD)$	**Mean** α_3 $(\pm 2SD)$
Blog-Net	0.34154 (±0.0068)	0.2963 (±0.0059)	0.36216 (±0.0024)
Flickr-Net	0.57146 (±0.0021)	0.22486 (±0.0082)	0.20368 (±0.0043)
Protein-Net	0.53127 (±0.0038)	0.24513 (±0.0029)	0.22360 (±0.0046)
Wiki-Net	0.26481 (±0.0029)	0.52432 (±0.0046)	0.21087 (±0.0018)
Cora-Net	0.19219 (±0.0037)	0.32149 (±0.0028)	0.48632 (±0.0043)
Cite-Net	0.27416 (±0.0018)	0.18256 (±0.0027)	0.54328 (±0.0034)

Table 4.2 indicates that for Blog-Net, the WeightedAvg ensemble model has combined the learnings of the base models in equal proportions. In Flickr-Net and Protein-Net, the weights of adjacency based similarity, multi-hop based similarity and random walk based similarity were $\sim 2:1:1$. In Wiki-Net the weights of adjacency based similarity, multi-hop based similarity and random walk based similarity were $\sim 1:2:1$. Finally, for Cora-Net and Cite-Net the weights of adjacency based similarity, multi-hop based similarity and random walk based similarity were $\sim 1:1:2$.

Using eigen-gap heuristic, graph of singular values of revealed that Blog-Net had five clusters (Figure 3.5b), Flickr-Net had three clusters (Figure 3.7b), Protein-Net had two clusters (Figure 3.9b), Wiki-Net had three clusters (Figure 3.11b), Cora-Net had three clusters (Figure 3.13b) and Cite-Net had five clusters (Figure 3.15b).

Table 4.3 gives the results obtained on the parameter separation index. Blog-Net, Flickr-Net and Wiki-Net have diam(G) in the range of $2-4$. This indicates that nodes are located close to each other. Whereas, for Protein-net, Cora-Net and Cite-Net the diam(G) is $6-8$ indicating the comparatively longer distance between nodes. Thus, the separation index of Blog-Net, Flickr-Net and Wiki-Net will have a higher range then Protein-net and Cora-Net. As Cite-Net has a larger number of clusters, its separation index shall have a higher range. The L.S.R. obtained using mean ensemble method were higher (+) or lower (-) on separation index compared to the closest technique as mentioned: Wiki-Net (-23.5%), Cora-Net (-9%), Cite-Net (-6%), Blog-Net (-15%), Flickr-Net (-17%) and Protein-Net (-52%). The L.S.R. obtained using

weighted-average ensemble method were higher (+) or lower (-) on separation index compared to the closest technique as mentioned: Wiki-Net (-4%), Cora-Net (4%), Cite-Net (5%), Blog-Net (-5%), Flickr-Net (-28%) and Protein-Net (-38%).

Table 4.3: Comparison of WeightedAvg, Mean with DeepWalk, LINE, MF on separation index

Data-set	**DeepWalk**	**LINE**	**MF**	**WeightedAvg**	**Mean**
Blog-Net	**3.89** (±0.16)	4.88 (±0.25)	5.23 (±0.46)	4.09 (±0.17)	4.38 (±0.07)
Flickr-Net	2.57 (±0.24)	**2.25** (±0.34)	3.08 (±0.19)	2.91 (±0.21)	2.71 (±0.04)
Protein-Net	1.85 (±0.27)	**1.36** (±0.12)	1.82 (±0.18)	1.73 (±0.07)	2.02 (±0.06)
Wiki-Net	2.85 (±0.31)	3.27 (±0.29)	3.19 (±0.28)	3.02 (±0.12)	**2.65** (±0.14)
Cora-Net	2.21 (±0.16)	2.59 (±0.31)	2.82 (±0.19)	**2.06** (±0.13)	2.43 (±0.04)
Cite-Net	4.23 (±0.58)	4.58 (±0.41)	5.07 (±0.49)	**4.02** (±0.21)	4.64 (±0.26)

Table 4.4 gives the results obtained on the parameter widest within-cluster gap. The L.S.R. obtained using mean ensemble method were higher (+) or lower (-) on widest within-cluster gap compared to closest technique as mentioned: Flickr-Net (-2%), Wiki-Net (0.2%), Cite-Net (22%), Blog-Net (-32%), Protein-Net (-12%) and Cora-Net (-31%). The L.S.R. obtained using weighted-average ensemble method were higher (+) or lower (-) on widest within-cluster gap compared to closest technique as mentioned: Flickr-Net (5%), Wiki-Net (-5%), Cite-Net (11%), Blog-Net (-21%), Protein-Net (-5%) and Cora-Net (-22%).

Table 4.4: Comparison of WeightedAvg, Mean with DeepWalk, LINE, MF on widest within-cluster gap

Data-set	**DeepWalk**	**LINE**	**MF**	**WeightedAvg**	**Mean**
Blog-Net	2.68 (±0.14)	1.85 (±0.16)	**3.23** (±0.34)	2.73 (±0.21)	2.53 (±0.11)
Flickr-Net	5.62 (±0.31)	6.20 (±0.56)	5.87 (±0.18)	**6.54** (±0.59)	6.08 (±0.21)
Protein-Net	**8.92** (±0.76)	7.65 (±0.68)	8.82 (±0.46)	8.39 (±0.37)	7.89 (±0.29)
Wiki-Net	6.45 (±0.48)	5.68 (±0.51)	6.48 (±0.49)	6.17 (±0.37)	**6.53** (±0.57)
Cora-Net	5.93 (±0.46)	6.59 (±0.61)	**8.22** (±0.78)	6.85 (±0.32)	6.36 (±0.44)
Cite-Net	2.09 (±0.18)	2.59 (±0.31)	1.59 (±0.17)	2.88 (±0.27)	**3.02** (±0.3)

Table 4.5 gives the results obtained on the parameter average silhouette width. The

L.S.R. obtained using mean ensemble method were higher (+) or lower (-) on average silhouette width compared to closest technique as mentioned: Blog-Net (9%), Cora-Net (-2%), Cite-Net (-5%), Flickr-Net (-6%), Protein-Net (-3%) and Wiki-Net (-11%). The L.S.R. obtained using weighted-average ensemble method were higher (+) or lower (-) on average silhouette width compared to closest technique as mentioned: Blog-Net (5%), Cora-Net (2%), Cite-Net (3%), Flickr-Net (-12%), Protein-Net (-1%) and Wiki-Net (-16%).

Table 4.5: Comparison of WeightedAvg, Mean with DeepWalk, LINE, MF on average silhouette width

Data-set	**DeepWalk**	**LINE**	**MF**	**WeightedAvg**	**Mean**
Blog-Net	0.03 (±0.0011)	-0.13 (±0.0016)	0.19 (±0.0046)	0.24 (±0.0027)	**0.28** (±0.0019)
Flickr-Net	0.19 (±0.0022)	**0.31** (±0.0037)	0.26 (±0.0028)	0.19 (±0.0016)	0.25 (±0.002)
Protein-Net	0.46 (±0.0024)	**0.47** (±0.0031)	**0.47** (±0.0036)	0.46 (±0.0021)	0.44 (±0.0019)
Wiki-Net	0.37 (±0.0042)	0.33 (±0.0026)	**0.42** (±0.0034)	0.26 (±0.0016)	0.31 (±0.0021)
Cora-Net	0.46 (±0.0025)	0.43 (±0.0037)	-0.21 (±0.0019)	**0.48** (±0.0036)	0.44 (±0.001)
Cite-Net	0.13 (±0.0011)	0.21 (±0.0016)	-0.06 (±0.009)	**0.24** (±0.0017)	0.16 (±0.0015)

Table 4.6 gives the results obtained on the parameter average distance between clusters. The L.S.R. obtained using mean ensemble method were higher (+) or lower (-) on average distance between clusters compared to closest technique as mentioned: Flickr-Net (8%), Wiki-Net (6%), Cora-Net (14%), Cite-Net (-8%), Blog-Net (-43%) and Protein-Net (-15%). The L.S.R. obtained using weighted-average ensemble method were higher (+) or lower (-) on average distance between clusters compared to closest technique as mentioned: Flickr-Net (-5%), Wiki-Net (10%), Cora-Net (8%), Cite-Net (4%), Blog-Net (-52%) and Protein-Net (-11%).

Table 4.6: Comparison of WeightedAvg, Mean with DeepWalk, LINE, MF on average distance between clusters

Data-set	**DeepWalk**	**LINE**	**MF**	**WeightedAvg**	**Mean**
Blog-Net	2.08 (±0.24)	0.49 (±0.089)	**1.83** (±0.15)	0.93 (±0.13)	1.08 (±0.09)
Flickr-Net	2.39 (±0.27)	2.82 (±0.54)	3.08 (±0.21)	2.96 (±0.14)	**3.22** (±0.16)
Protein-Net	4.97 (±0.38)	**6.82** (±0.59)	5.82 (±0.64)	6.28 (±0.57)	5.91 (±0.37)
Wiki-Net	2.85 (±0.34)	3.74 (±0.29)	3.18 (±0.42)	**4.12** (±0.36)	3.94 (±0.41)
Cora-Net	2.94 (±0.27)	3.07 (±0.25)	1.85 (±0.11)	3.25 (±0.18)	**3.56** (±0.31)
Cite-Net	0.82 (±0.18)	1.28 (±0.25)	0.79 (±0.037)	**1.32** (±0.14)	1.19 (±0.09)

Table 4.7 gives the results obtained on the parameter Dunn index. The L.S.R. obtained using mean ensemble method were higher (+) or lower (-) on Dunn index compared to the closest technique as mentioned: Protein-Net (16%), Cora-Net (-12%), Blog-Net (18%), Flickr-Net (36%), Wiki-Net (2%) and Cite-Net (30%). The L.S.R. obtained using weighted-average ensemble method were higher (+) or lower (-) on Dunn index compared to the closest technique as mentioned: Protein-Net (-8%), Cora-Net (-3%), Blog-Net (12%), Flickr-Net (14%), Wiki-Net (7%) and Cite-Net (11%).

Table 4.7: Comparison of WeightedAvg, Mean with DeepWalk, LINE, MF on Dunn index

Data-set	**DeepWalk**	**LINE**	**MF**	**WeightedAvg**	**Mean**
Blog-Net	0.26 (±0.006)	**0.11** (±0.0042)	0.32 (±0.0027)	0.23 (±0.0034)	0.29 (±0.0013)
Flickr-Net	**0.21** (±0.0028)	0.41 (±0.0031)	0.31 (±0.0046)	0.35 (±0.0038)	0.57 (±0.0028)
Protein-Net	0.28 (±0.0031)	0.21 (±0.0037)	0.42 (±0.0051)	**0.13** (±0.0036)	0.37 (±0.0021)
Wiki-Net	0.39 (±0.0019)	**0.27** (±0.0024)	0.47 (±0.0031)	0.34 (±0.003)	0.29 (±0.0018)
Cora-Net	0.46 (±0.0024)	0.61 (±0.0028)	0.52 (±0.0031)	0.43 (±0.0019)	**0.34** (±0.001)
Cite-Net	**0.13** (±0.0054)	0.26 (±0.0037)	0.34 (±0.0061)	0.24 (±0.0059)	0.43 (±0.0037)

4.5 Discussion of Results and Summary

Network representation learning has emerged as a preferred method of transforming social network data to make it more amenable to analysis. Multiple categories of N.R.L. techniques are available in the literature; however, there does not exist a single approach that could be suitable for all types of data-sets. A reason for this could be that each

category of technique has a single objective function. Hence, in this inquiry, N.R.L. was approached as a multi-objective optimization process. The resultant ensemble model was applied on data-sets with nodes $> 10^3$.

Blog-Net had Gini index $G = 0.39$ and clustering coefficient $c = 0.08$. This indicated the presence of more hubs (popular nodes) in the network. This conclusion is consistent with the observation of five clusters in the network of which three are well-separated (Figure 3.5b). Each hub has a large number of followers $d = 66.3$, and the value of c indicates that followers did not form relationships with each other. Such networks, when represented as vector embeddings, would lead to popular nodes and their followers occupying disjoint regions in the latent space. In such a scenario, any proximity preserving N.R.L. technique would be efficient. The WeightAvg model assigns $\sim 1 : 1 : 1$ weights to its base models and its performance resembles that of the mean ensemble model for this scenario. Flickr-Net and Protein-Net have a high Gini index $G = 0.63 - 0.67$ and low clustering co-efficient $0.09 - 0.1$. This indicates the presence of dominant nodes in the network. The graph of singular values for both networks (Figure 3.7b and Figure 3.9b) also indicates well-separated regions in the network. In such a scenario, the adjacency preserving technique may achieve effective results. WeightedAvg ensemble was observed to assign weights of adjacency based similarity, multi-hop based similarity and random walk based similarity in proportion of $\sim 2 : 1 : 1$. Wiki-Net data-set has high transitivity $c = 0.43$, it also has presence of dominant nodes $G = 0.62$. The graph of singular values (Figure 3.11b) indicates three well-separated clusters in the network. As $c = 0.43$ is higher, the second-order proximity between nodes (neighborhood overlap) could have a higher impact on the L.S.R. Therefore, in Wiki-Net the weights of adjacency based similarity, multi-hop based similarity and random walk based similarity were $\sim 1 : 2 : 1$. Cora-Net and Cite-Net have low diameter and average path length $\text{diam}(G) = 6-8$ and $\bar{P} = 1.74-1.81$. There are few popular nodes in the network $G =\sim 0.42$. Due to low diameter, a random walk on the nodes would traverse more nodes and effectively represent them in the latent space. Hence, for Cora-Net and Cite-Net, the weights of adjacency based similarity, multi-hop based similarity and random walk based similarity were $\sim 1 : 1 : 2$.

Table 4.8 gives the results of transitivity on the data-sets. The L.S.R. obtained us-

ing mean ensemble method were higher (+) or lower (-) on transitivity compared to the closest technique as mentioned: Protein-Net (3%), Cora-Net (5%), Blog-Net (-5%), Flickr-Net (4%), Wiki-Net (2%) and Cite-Net (1%). The L.S.R. obtained using weighted-average ensemble method were higher (+) or lower (-) on transitivity compared to the closest technique as mentioned: Protein-Net (-1%), Cora-Net (-3%), Blog-Net (2%), Flickr-Net (2%), Wiki-Net (-1%) and Cite-Net (-4%).

Table 4.8: Comparison of WeightedAvg, Mean with DeepWalk, LINE, MF on transitivity

Data-set	**Actual** c	**DeepWalk**	**LINE**	**MF**	**WeightedAvg**	**Mean**
Blog-Net	0.08	0.11 (±0.0042)	0.12 (±0.0068)	0.09 (±0.0037)	0.11 (±0.0034)	0.14 (±0.0028)
Flickr-Net	0.1	0.14 (±0.0061)	0.13 (±0.0054)	0.06 (±0.0038)	0.09 (±0.0043)	0.1 (±0.0033)
Protein-Net	0.09	0.12 (±0.0063)	0.06 (±0.0057)	0.08 (±0.0061)	0.07 (±0.0031)	0.12 (±0.0028)
Wiki-Net	0.43	0.27 (±0.0056)	0.46 (±0.0051)	0.34 (±0.0049)	0.39 (±0.0029)	0.38 (±0.0031)
Cora-Net	0.09	0.12 (±0.0037)	0.19 (±0.0053)	0.16 (±0.0027)	0.15 (±0.003)	0.14 (±0.0032)
Cite-Net	0.13	0.09 (±0.0012)	0.16 (±0.0024)	0.14 (±0.0006)	0.08 (±0.0012)	0.13 (±0.0004)

A key drawback of ensemble techniques was the computation complexity. The ensemble techniques have a computation complexity of $O(|E|d) + O(|V|d) + O(|E|d)$. This is a summation of the computation complexities of the base models. Constant time is required for performing the combination by consensus operation, i.e., weighted-average or mean.

N.R.L. techniques require that all nodes in the the graph should be present during the training of the encoder. Hence, these approaches are inherently transductive and do not naturally generalize to unseen nodes (non-inductive). Hence, “deep encoder" based techniques are needed that map the nodes to their embedding vectors using a complex non-linear mapping function. As the parameters used by the “deep encoder" based frameworks for mapping the nodes to their embedding vectors are shared, these techniques are more efficient compared to the “shallow encoder" based framework.

Chapter 5

Exploring Convolutional Auto-Encoders for Representation Learning on Networks

5.1 Introduction

Learning vector space embeddings of graph-structured or relational data are similar to embedding complex data into low-dimensional geometries. For an embedding method to be successful, this latent geometry must go beyond simple compression and provide a strong inductive bias for reasoning about the relationships between objects. After a latent geometry has been inferred, it is possible to represent the objects using their corresponding embeddings, and most importantly, these embeddings provide a powerful and natural feature space for these objects, which can then be used for network analysis models.

However, the model seen previously were shallow encoder based techniques. Network embedding techniques based on shallow encoders rely on the concept of “similarity" to learn embeddings of the nodes. This, in contrast with “deep encoder" based techniques that map the nodes to their embedding vectors using a complex non-linear mapping function. As the parameters used by the “deep encoder" based frameworks for mapping the nodes to their embedding vectors are shared, these techniques are more efficient

compared to the “shallow encoder" based framework. The current chapter focuses on models that learn the notion of “similarity" based on the data. This is done utilizing an encoder function that is trained to learn a complex non-linear mapping from input space to target space.

Deep learning architectures in the literature are based on Graph Convolutional Networks (G.C.N.). G.C.N. and its variants focus on information aggregation. A drawback of these models is the fully connected layer used in them increases the number of parameters. As the graph size increases, the number of parameters also increases, this leads to slower convergence of gradient descent. Therefore the current inquiry proposes two auto-encoder architectures which modify existing techniques like Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.) by replacing the fully connected layers with 1-D convolutional ones.

These architectures aim to maintain computational and storage efficiency, less number of parameters, localization and applicability to inductive problems. Additionally, an equivalence is demonstrated between the aggregation and transformation operations performed in the hidden layers of an auto-encoder and laplacian smoothing operation. Thus, a graph convolutional network architecture with multiple hidden layers would be similar to the repeated application of Laplacian smoothing on the data. This results in embeddings of data-points having similar features in the high dimension space to be closer in low dimension space. In addition to this, the representation vectors of the data-points obtained with the proposed auto-encoders were also observed to have better clustering quality. Extensive experiments were performed on network data-sets to validate these approaches.

5.1.1 Motivation

Deep learning has brought a paradigm shift in representation learning due to a combination of factors such as availability of voluminous data, cheap processing power and better optimization techniques. These methods [34, 35, 36, 37, 38] have achieved results close to state-of-the-art N.R.L. techniques.

5.1.2 Contribution

The main contribution in this chapter is to propose fully convolutional implementations of two standard G.C.N. architectures viz. Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.). The proposed architectures are called Graph Auto-Encoder with Convolutional layers (G.A.E.-F.C.) and Variational Graph Auto-Encoder with Convolutional layers (V.A.E.-F.C.) for latent space representation of networks. The advantages of these approaches are demonstrated over the standard Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.) frameworks by performing experiments on network data-sets.

The rest of the paper is organized as follows: In Section 5.2 a background of G.C.N. is provided, Section 5.3 describes the proposed method and Section 5.5 discusses the results. The conclusion of the inquiry is provided in Section 5.6.

5.2 Background of Graph Convolutional Networks

A graph $G(V, E)$ with V as the node-set and E as the dyad set has a binary adjacency matrix $A \in \mathbb{R}^{|V|*|V|}$ and a node feature matrix $X \in \mathbb{R}^{|V|*m}$ [35]. The node feature matrix is formed by stacking the m-dimension feature vector of each node horizontally. G.C.N. has the following steps:

1. The convolutional filter $\hat{A}$ is constructed from the new adjacency matrix $\tilde{A} = A+I$ and new degree matrix $\tilde{D} = D + I$ by the operation $\hat{A} = \tilde{D}^{\frac{-1}{2}}\tilde{A}\tilde{D}^{\frac{-1}{2}}$

2. A propagation rule for the graph convolution layer is defined as $H^{(1)} = (\hat{A}H^{(0)}\theta^{(0)})$. $H^{(1)}$ is matrix of activations of first layer, $H^{(0)} = X$, $\theta^{(0)}$ is a trainable weight matrix of layer 0 and is a non-linear activation function.

3. The convolved representations of the vertices $\hat{A}H^{(0)}$ are fed to a standard fully connected layer as given in Figure 5.1.

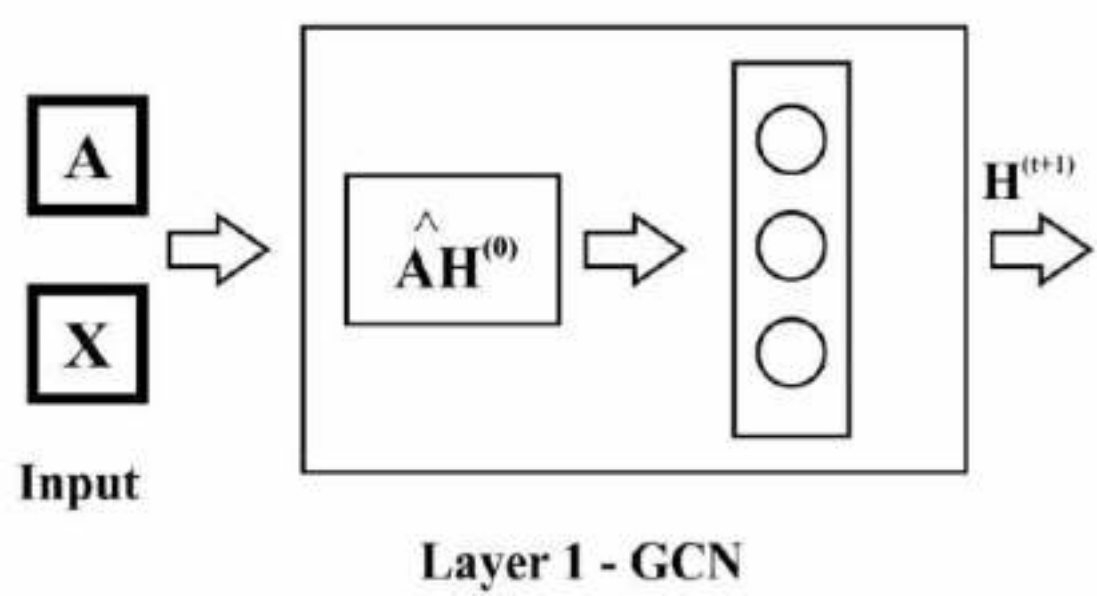


Figure 5.1: Graph Convolutional Network

5.3 Graph Auto-Encoder Network

Graph Auto-encoder proposed by T Kipf *et al.* takes as input the adjacency matrix A of the entire graph and the node feature matrix X. It outputs the reconstructed adjacency matrix $\hat{A}$ and the node embedding matrix Z. The loss function during learning is reconstruction loss which is calculated as $L = E_{q(Z|X,A)}[logp(A|Z)]$. An inner product function is used as the decoder $\hat{A} = \sigma(ZZ^T)$. As the encoder function minimizes the loss due to reconstruction error, the representations of nodes preserve first-order proximity.

Figure 5.2 provides the architecture of the encoder of the G.A.E. with 1-D fully convolutional (F.C.) layer in place of the fully connected (dense) layer. These F.C. encoder blocks are stacked to obtain the encoder of the G.A.E. The inner product decoder is kept the same.

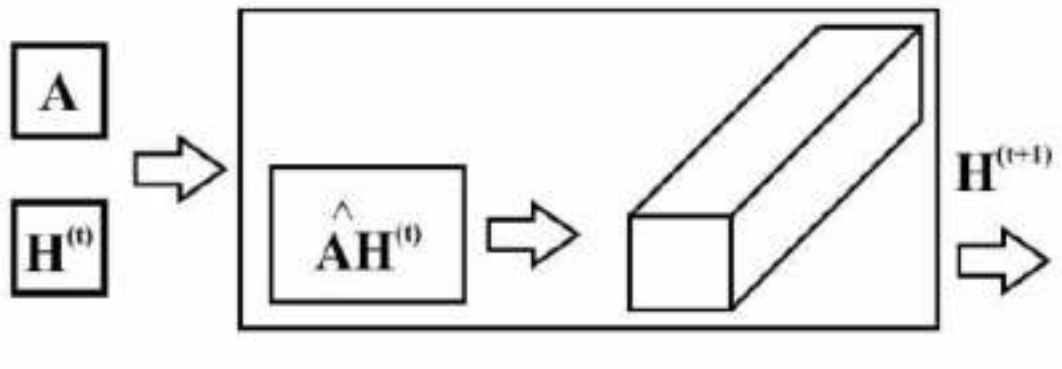


Figure 5.2: Architecture of Graph Auto-Encoder with Fully Convolutional layer (G.A.E.-F.C.)

5.4 Variational Graph Auto-Encoder Network

The Variational Graph Auto-Encoder (V.A.E.) proposed by T Kipf *et al.* [121] infers the latent variables Z by sampling from a normal distribution. The μ, σ^2 of this normal distribution is generated through a two-layer GCN as $\mu = GCN_\mu(X, A), \sigma^2 = GCN_\sigma(X, A), GCN(X, A) = \hat{A}ReLU(\hat{A}X\theta^{(0)})\theta^{(1)}$. Then for each input x_i, $q(Z_i|X, A) = N(Z_i|\mu_i, diag({\sigma_i}^2))$. The decoder is same as the inner product function of G.A.E. to obtain the reconstructed adjacency matrix $A' = \sigma(Z.Z^T)$. The loss function has two parts as given in Eq 5.1: First part is the reconstruction loss to ensure reconstructed adjacency matrix A' is similar to input one A. The second part is KL-divergence to ensure distribution of latent variable is close to a normal distribution. The V.A.E. model depends on the G.C.N. and hence due to the fully connected layers, as the graph size increases, the number of parameters also increase, this leads to slower convergence of gradient descent.

$$L = E_{q(Z|X,A)}[logp(A|Z)] - KL(q(Z|X, A)||p(Z)) \tag{5.1}$$

where, $p(Z) = \prod_i p(z_i) = \prod_i N(z_i|0, I)$

Similar to the 1-D convolutional Graph Auto-encoder, an F.C. encoder block is used to

substitute the fully connected layer in the V.A.E. as given in Figure 5.3.

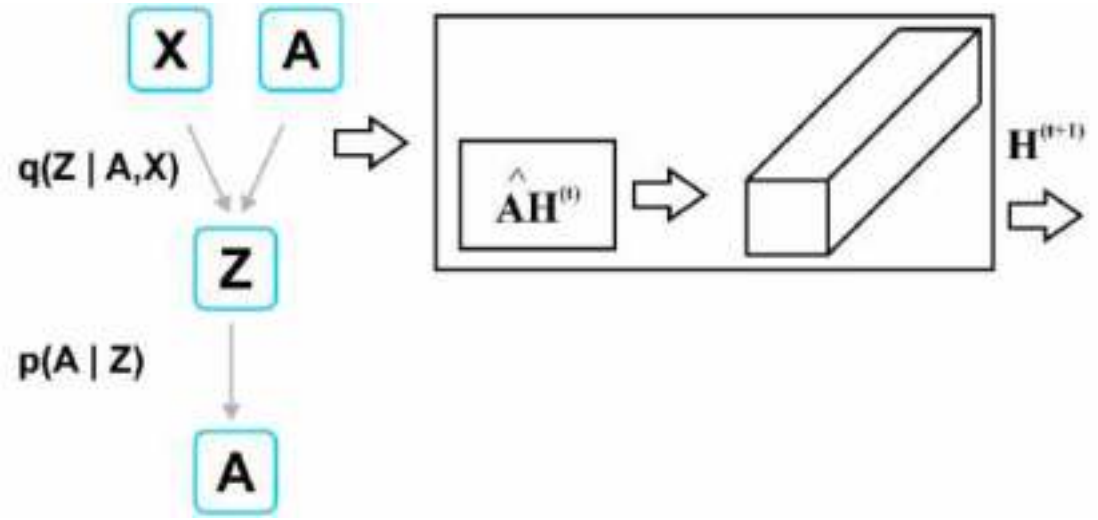


Figure 5.3: Architecture of Variational Graph Auto-encoder with Fully convolutional layer (VAE-FC)

5.4.1 Understanding neighborhood-aggregation encoder algorithm of G.A.E., V.A.E., G.A.E.-F.C. and V.A.E.- F.C.

As given in Algorithm 5, the encoders of G.A.E., V.A.E., G.A.E.-F.C. and V.A.E.-F.C. use the neighborhood aggregation strategy. There are two operations: AGGREGATE and COMBINE. AGGREGATE operation is common for G.A.E., V.A.E., G.A.E.-F.C. and V.A.E.- F.C. The difference between convolutional implementations (G.A.E.-F.C. and V.A.E.- F.C) and other implementations (G.A.E. and V.A.E.) is in the COMBINE operation. The convolutional implementations use 1-D C.N.N. in the COMBINE operation wheres, the other implementations utilize a fully connected layer.

The complexity analysis, for an input graph having N nodes and feature vectors of length C, the adjacency matrix A will be in R^{N*N} and input feature matrix V^a will be of size R^{N*C}. For a given graph convolution layer, to obtain an output $V^{out} = R^{N*F}$, the time complexity is $O(N^2CF)$. A convolution operation with kernel size R^{p*p} in a conventional C.N.N., operating on the same input has time complexity of $O(Np)$.

Algorithm 5: Neighborhood aggregation encoder algorithm of G.A.E., V.A.E., G.A.E.-F.C. and V.A.E.- F.C.. Adapted from [39]

Inputs: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\{\mathbf{W}^k, \forall k \in [1, K]\}$; non-linearity σ; differentiable aggregator functions $\{\text{AGGREGATE}_k, \forall k \in [1, K]\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Outpt : Vector embedding $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;

for $k = 1...K$ **do**

  for $v \in \mathcal{V}$ **do**

    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;

    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$

  end

  $\mathbf{h}_v^k \leftarrow \text{NORMALIZE}(\mathbf{h}_v^k), \forall v \in \mathcal{V}$

end

$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

5.4.2 Advantages of Fully Convolutional layers

- Fixed parameters irrespective of the input size of the graph
- Fast convergence on gradient descent compared to fully connected layers
- Fewer parameters than fully connected layers

The two models proposed in Sections 5.3 and 5.4 are trained to learn embeddings using the procedure given in Algorithm 6.

Algorithm 6: Model Training and Fitting

Data: X, A

Result: Z

1 Initialization of encoder weights θ and bias b ;
2 Configure model to minimize Loss in Eq. 5.1;
3 Add activation function: rectified linear unit to encoder;
4 Create inner product decoder layer function;
5 Set epochs, batch size;
6 Optimize the loss function L using gradient descent;

5.4.3 Relation of auto-encoders with laplacian smoothing

Both the proposed auto-encoder models perform operations that are equivalent to a laplacian smoothing operation. Consider a curve with points p_i, p_{i-1}, p_{i+1}.. as shown in Figure 5.4 where laplacian smoothing is applied to bring each point closer to weighted average of its neighbors using Eq. 5.2 and 5.3.

$$p_i \leftarrow p_i + \frac{1}{2}L(p_i) \tag{5.2}$$

$$L(p_i) = \frac{p_{i+1} + P_{i-1}}{2} = p_i \tag{5.3}$$

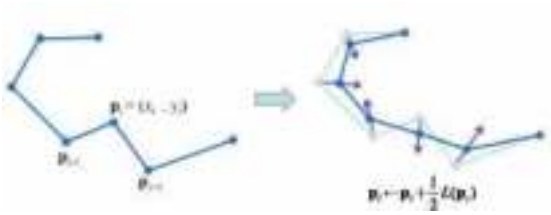


Figure 5.4: Example of laplacian smoothing where a curve is shown before and after the smoothing operation is applied [122]

For a graph $G = (V, E)$ with adjacency matrix A such that $a_{ij} = 1, |i - j| = 1$ and degree matrix $D = diag(d_1, d_2, .., d_n)$, the smoothing operation can be re-written as,

$$P \leftarrow (I - \frac{1}{2}L_{rw})P \tag{5.4}$$

where,

$$P = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix} \quad L_{\text{rw}} = \begin{bmatrix} 1 & -1 & & & \\ -\frac{1}{2} & 1 & -\frac{1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{1}{2} & 1 & -\frac{1}{2} \\ & & & -1 & 1 \end{bmatrix} \tag{5.5}$$

The matrix $L = D - A$ is the normalized graph Laplacian and has two versions $L_{sym} = D^{-1/2}LD^{-1/2}$ and $L_{rw} = D^{-1}L$. The laplacian smoothing operation is $P \leftarrow (I - \gamma L_{rw})P$ and $0 \leq \gamma \leq 1$ controls strength of smoothing. Setting $\gamma = 0$, laplacian smoothing becomes equivalent to non-linear mapping function (identity function) being learned by Auto-encoders. The Laplacian smoothing computes the local average of each vertex as its new representation. Since vertices in the same cluster tend to be densely connected, the smoothing makes their features similar, which makes the subsequent representation learning task amenable.

5.5 Experimental Study

5.5.1 Data-sets

Network data-sets with $|V| > 10^3$ were chosen for the evaluation of the proposed deep learning based encoders technique viz. G.A.E.-F.C. and V.A.E.-F.C.

Table 5.1: Description of Network Data-sets with binary attributes

Description	**Blog-Net**	**Flickr-Net**	**Protein-Net**	**Wiki-net**	**Cora-net**	**Cite-net**
$\|V\|$	5196	7575	3890	4777	2708	3312
V^a	8189	12047	50	40	1434	3704
$\|E\|$	171743	239738	37845	54810	5429	4732
c	0.08	0.1	0.09	0.43	0.14	0.13

5.5.2 Encoder architectures

Encoder architectures of the Graph Auto-Encoder (G.A.E.), Variational Graph Auto-Encoder (V.A.E.), Graph Auto-Encoder with Convolutional layers (G.A.E.-F.C.) and

Variational Graph Auto-Encoder with Convolutional layers (V.A.E.-F.C.) are described in Table 5.2, Table 5.3, Table 5.4 and Table 5.5 respectively. T Kipf *et al.* **??** implementation of G.A.E. and V.A.E is used. G.A.E.-F.C. and V.A.E.-F.C. are built using Keras package in Python [123].

Table 5.2: Dimensions of the dense layers in G.A.E encoder

layer name	Blog-Net	Cora-Net	Cite-Net	Flickr-Net	Protein-Net	Wiki-Net
X	5196*8189	2708*1434	3312*3704	7575*12047	3890*50	4777*40
A	5196*5196	2708*2708	3312*3312	7575*7575	3890*3890	4777*4777
dense_encode	8189*32	1434*32	3704*32	12047*32	50*32	40*32

Table 5.3: Dimensions of the dense layers in V.A.E. encoder

layer name	Blog-Net	Cora-Net	Cite-Net	Flickr-Net	Protein-Net	Wiki-Net
X	5196*8189	2708*1434	3312*3704	7575*12047	3890*50	4777*40
A	5196*5196	2708*2708	3312*3312	7575*7575	3890*3890	4777*4777
dense_encode	8189*64	1434*64	3704*64	12047*64	50*32	40*32
dense_encode	64*32	64*32	64*32	64*32	32*32	32*32

Table 5.4: Dimensions of the 1-D convolution layers in G.A.E-F.C. encoder

layer name	Blog-Net	Cora-Net	Cite-Net	Flickr-Net	Protein-Net	Wiki-Net
X	5196*8189	2708*1434	3312*3704	7575*12047	3890*50	4777*40
A	5196*5196	2708*2708	3312*3312	7575*7575	3890*3890	4777*4777
conv_encode	1x8158	1x1403	1x3673	1x12016	1x19	1x9
stride	1	1	1	1	1	1
height	1	1	1	1	1	1
padding	valid	valid	valid	valid	valid	valid

Table 5.5: Dimensions of the 1-D convolution layer of V.A.E.-F.C. encoder

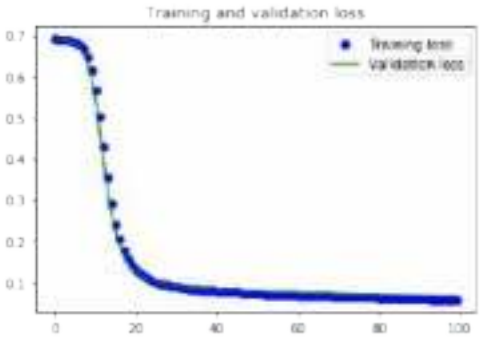
layer name	Blog-Net	Cora-Net	Cite-Net	Flickr-Net	Protein-Net	Wiki-Net
X	5196*8189	2708*1434	3312*3704	7575*12047	3890*50	4777*40
A	5196*5196	2708*2708	3312*3312	7575*7575	3890*3890	4777*4777
conv_encode	1x8126	1x1371	1x3641	1x11984	1x19	1x9
stride	1	1	1	1	1	1
height	1	1	1	1	1	1
padding	valid	valid	valid	valid	valid	valid
conv_encode	1x33	1*33	1x33	1x33	1*1	1x1
stride	1	1	1	1	1	1
height	1	1	1	1	1	1
padding	valid	valid	valid	valid	valid	valid

The architectures convert network data of Blog-Net, Cora-Net, Cite-Net, Flickr-Net, Protein-Net and Wiki-Net from their original dimensions into vector space of dimension = 32. The quality of the vertex embeddings is measured using distance-based statistics obtained from clustering the representations. The parameters are separation index, widest within-cluster gap, average silhouette width, the average distance between clusters and Dunn index. Twenty iterations are performed on vertex embeddings for calculating each performance metric. The values of mean along with limits of two standard deviations $2SD$ of the mean (upper and lower bounds of the 95% confidence interval) are given for quantitative comparison. Additionally, the architectures are compared on the basis of their parameters, time per iteration and iterations required for convergence on the data-sets.

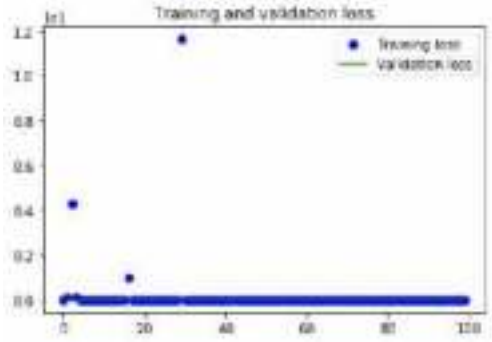
5.5.3 Results

The two variants of the auto-encoder architecture are trained as per the Algorithm 6 to generate vector embeddings. The embeddings are then applied to node clustering to test the efficacy of the proposed models. Standard performance metrics described in the previous chapter viz. Separation index, Widest within-cluster gap, Average silhouette width, Average distance between clusters and Dunn index are used to compare the clustering results. Results of a single iteration are given below.

Figure 5.5 shows binary cross-entropy loss (X-axis) for different epochs (Y-axis) on the training and test set of Blog-Net using Graph Auto-Encoder (G.A.E.) (Figure 5.5a), Variational Graph Auto-Encoder (V.A.E.) (Figure 5.5b), Graph Auto-Encoder with Convolutional layers (G.A.E.-F.C.) (Figure 5.5c) and Variational Graph Auto-Encoder with Convolutional layers (V.A.E.-F.C.) (Figure 5.5d).



(a) Binary cross-entropy loss on training and test set of Blog-Net using G.A.E.



(b) Binary cross-entropy loss on training and test set of Blog-Net using V.A.E.

(c) Binary cross-entropy loss on training and test set of Blog-Net using G.A.E.-F.C.

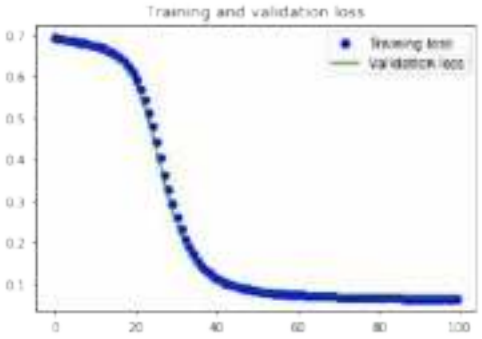
(d) Binary cross-entropy loss on training and test set of Blog-Net using V.A.E.-F.C.

Figure 5.5: Binary cross-entropy loss on training and test set of Blog-Net

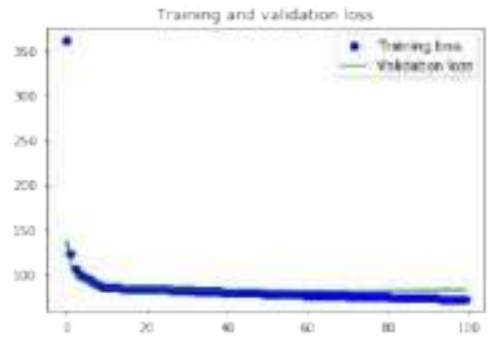
Figure 5.5a shows that loss on the test set converges to 0.012 after 96 iterations. Figure 5.5b shows that loss on the test set converges to 81 after 36 iterations. Figure 5.5c shows that loss on the test set is confined to range $0.08 - 0.1$ after 16 iterations. Figure 5.5d shows that loss on the test set decreases from $0.043 - 0.037$ with iterations.

Figure 5.6 shows binary cross-entropy loss (X-axis) for different epochs (Y-axis) on the training and test set of Cora-Net using Graph Auto-Encoder (G.A.E.) (Figure 5.6a),

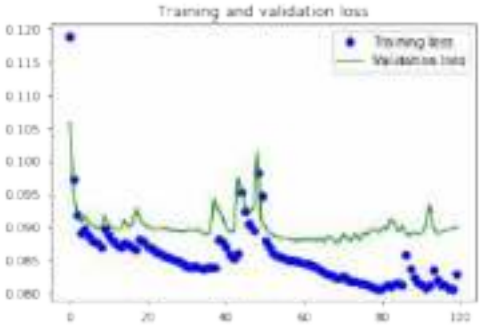
Variational Graph Auto-Encoder (V.A.E.) (Figure 5.6b), Graph Auto-Encoder with Convolutional layers (G.A.E.-F.C.) (Figure 5.6c) and Variational Graph Auto-Encoder with Convolutional layers (V.A.E.-F.C.) (Figure 5.6d).



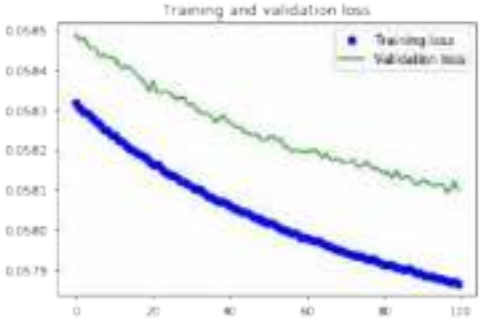
(a) Binary cross-entropy loss on training and test set of Cora-Net using G.A.E.



(b) Binary cross-entropy loss on training and test set of Cora-Net using V.A.E.



(c) Binary cross-entropy loss on training and test set of Cora-Net using G.A.E.-F.C.



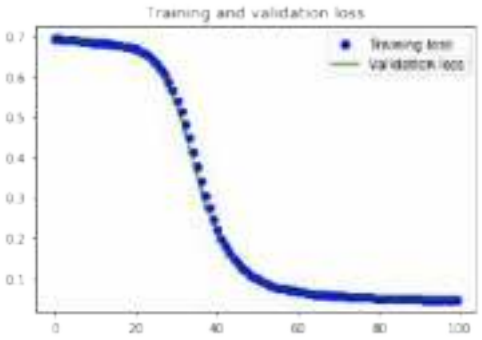
(d) Binary cross-entropy loss on training and test set of Cora-Net using V.A.E.-F.C.

Figure 5.6: Binary cross-entropy loss on training and test set of Cora-Net

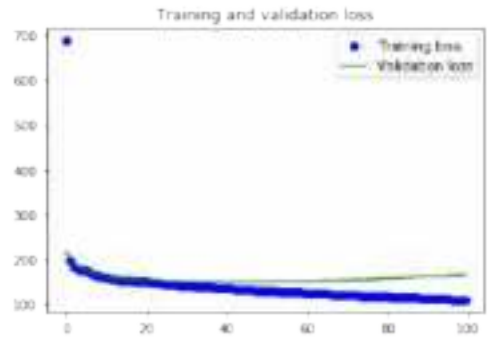
Figure 5.6a shows that loss on the test set converges to 0.012 after 42 iterations. Figure 5.6b shows that loss on the test set converges to 73 after 84 iterations. Figure 5.6c shows that loss on the test set is confined to range 0.09 – 0.8 after 23 iterations. Figure 5.6d shows that loss on the test set decreases from 0.0583 – 0.0579 with iterations.

Figure 5.7 shows binary cross-entropy loss (X-axis) for different epochs (Y-axis) on the training and test set of Cite-Net using Graph Auto-Encoder (G.A.E.) (Figure 5.7a), Variational Graph Auto-Encoder (V.A.E.) (Figure 5.7b), Graph Auto-Encoder with Convolutional layers (G.A.E.-F.C.) (Figure 5.7c) and Variational Graph Auto-Encoder

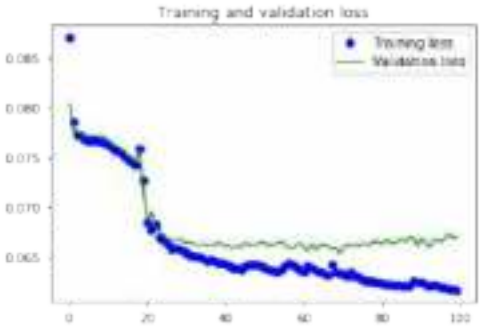
with Convolutional layers (V.A.E.-F.C.) (Figure 5.7d).



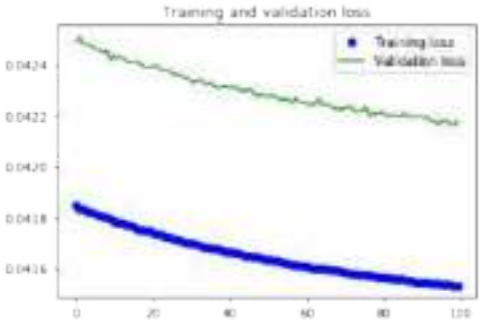
(a) Binary cross-entropy loss on training and test set of Cite-Net using G.A.E.



(b) Binary cross-entropy loss on training and test set of Cite-Net using V.A.E.



(c) Binary cross-entropy loss on training and test set of Cite-Net using G.A.E.-F.C.

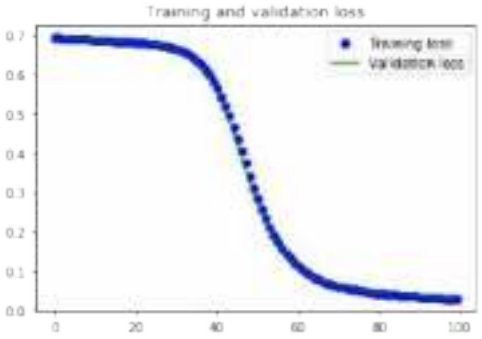


(d) Binary cross-entropy loss on training and test set of Cite-Net using V.A.E.-F.C.

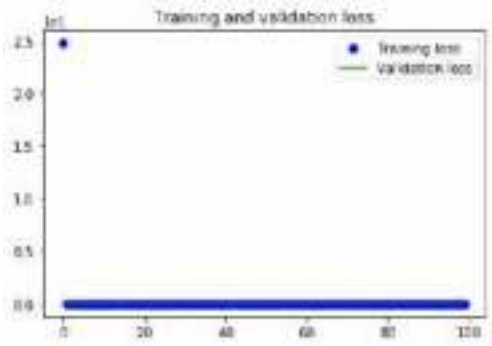
Figure 5.7: Binary cross-entropy loss on training and test set of Cite-Net

Figure 5.7a shows that loss on the test set converges to 0.008 after 92 iterations. Figure 5.7b shows that loss on the test set converges to 68 after 98 iterations. Figure 5.7c shows that loss on the test set is confined to range $0.08 - 0.65$ after 19 iterations. Figure 5.7d shows that loss on the test set decreases from $0.0424 - 0.0416$ with iterations.

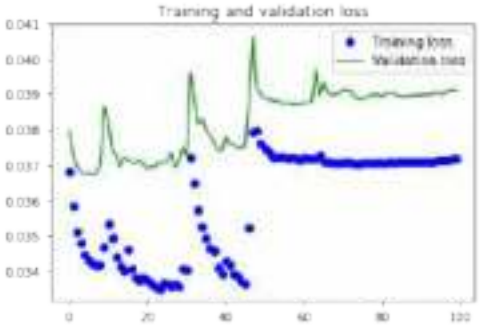
Figure 5.8 shows binary cross-entropy loss (X-axis) for different epochs (Y-axis) on the training and test set of Flickr-Net using Graph Auto-Encoder (G.A.E.) (Figure 5.8a), Variational Graph Auto-Encoder (V.A.E.) (Figure 5.8b), Graph Auto-Encoder with Fully Convolutional layers (G.A.E.-F.C.) (Figure 5.8c) and Variational Graph Auto-Encoder with Fully Convolutional layers (V.A.E.-F.C.) (Figure 5.8d).



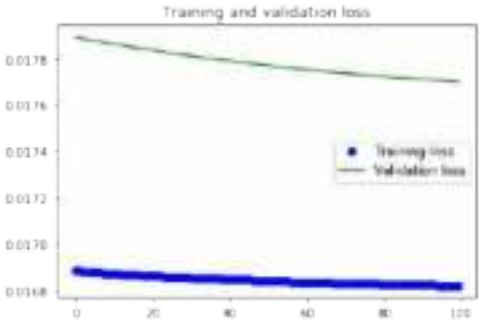
(a) Binary cross-entropy loss on training and test set of Flickr-Net using G.A.E.



(b) Binary cross-entropy loss on training and test set of Flickr-Net using V.A.E.



(c) Binary cross-entropy loss on training and test set of Flickr-Net using G.A.E.-F.C.

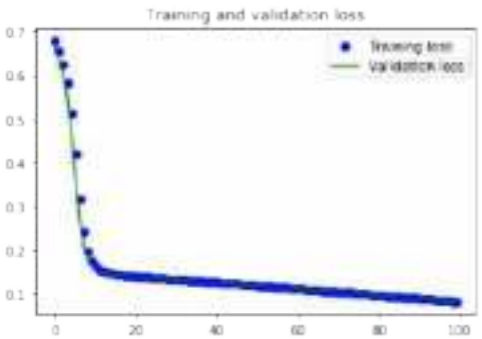


(d) Binary cross-entropy loss on training and test set of Flickr-Net using V.A.E.-F.C.

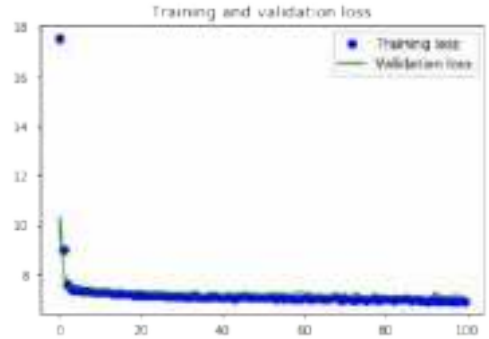
Figure 5.8: Binary cross-entropy loss on training and test set of Flickr-Net

Figure 5.8a shows that loss on the test set converges to 0.043 after 93 iterations. Figure 5.8b shows that loss on the test set converges to 629 after 98 iterations. Figure 5.8c shows that loss on the test set is confined to range $0.038 - 0.034$ after 23 iterations. Figure 5.8d shows that loss on the test set decreases from $0.0178 - 0.0176$ with iterations.

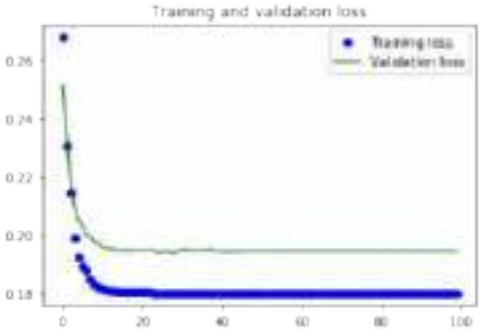
Figure 5.9 shows binary cross-entropy loss (X-axis) for different epochs (Y-axis) on the training and test set of Protein-Net using Graph Auto-Encoder (G.A.E.) (Figure 5.9a), Variational Graph Auto-Encoder (V.A.E.) (Figure 5.9b), Graph Auto-Encoder with Convolutional layers (G.A.E.-F.C.) (Figure 5.9c) and Variational Graph Auto-Encoder with Convolutional layers (V.A.E.-F.C.) (Figure 5.9d).



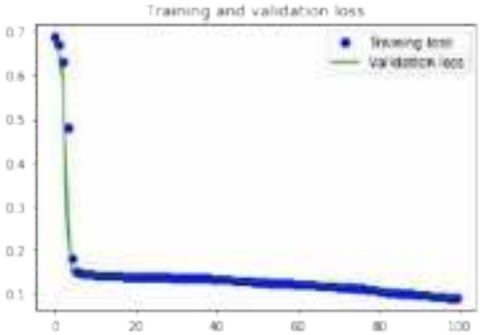
(a) Binary cross-entropy loss on training and test set of Protein-Net using G.A.E.



(b) Binary cross-entropy loss on training and test set of Protein-Net using V.A.E.



(c) Binary cross-entropy loss on training and test set of Protein-Net using G.A.E.-F.C.

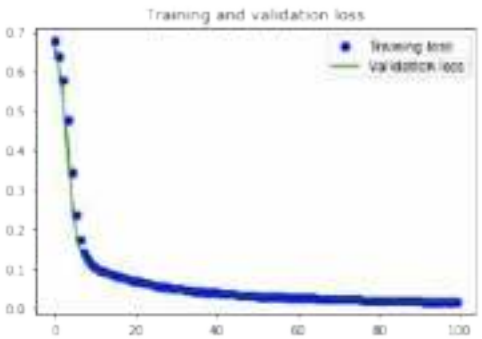


(d) Binary cross-entropy loss on training and test set of Protein-Net using V.A.E.-F.C.

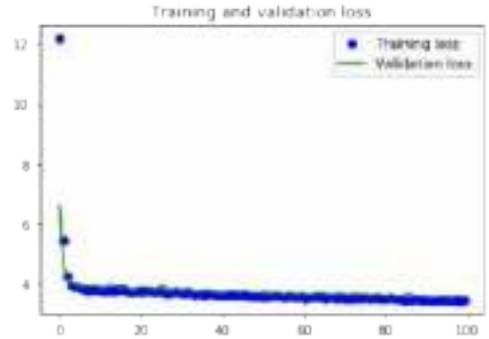
Figure 5.9: Binary cross-entropy loss on training and test set of Protein-Net

Figure 5.9a shows that loss on the test set converges to 0.008 after 94 iterations. Figure 5.9b shows that loss on the test set converges to 8 after 76 iterations. Figure 5.9c shows that loss on the test set is confined to range $0.21 - 0.18$ after 16 iterations. Figure 5.9d shows that loss on the test set decreases from $0.15 - 0.04$ with iterations.

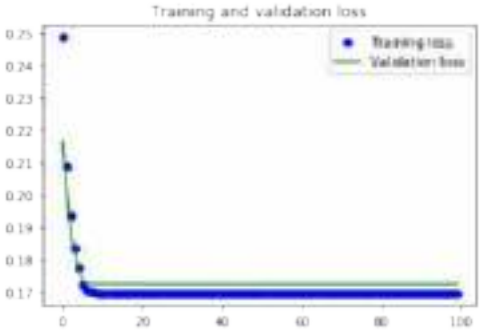
Figure 5.10 shows binary cross-entropy loss (X-axis) for different epochs (Y-axis) on the training and test set of Wiki-Net using Graph Auto-Encoder (G.A.E.) (Figure 5.10a), Variational Graph Auto-Encoder (V.A.E.) (Figure 5.10b), Graph Auto-Encoder with Convolutional layers (G.A.E.-F.C.) (Figure 5.10c) and Variational Graph Auto-Encoder with Convolutional layers (V.A.E.-F.C.) (Figure 5.10d).



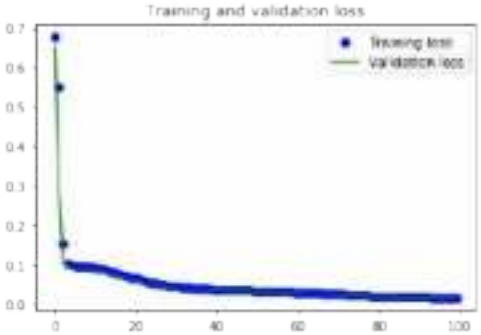
(a) Binary cross-entropy loss on training and test set of Wiki-Net using G.A.E.



(b) Binary cross-entropy loss on training and test set of Wiki-Net using V.A.E.



(c) Binary cross-entropy loss on training and test set of Wiki-Net using G.A.E.-F.C.



(d) Binary cross-entropy loss on training and test set of Wiki-Net using V.A.E.-F.C.

Figure 5.10: Binary cross-entropy loss on training and test set of Wiki-Net

Figure 5.10a shows that loss on the test set converges to 0.014 after 74 iterations. Figure 5.10b shows that loss on the test set converges to 4.2 after 36 iterations. Figure 5.10c shows that loss on the test set is confined to range 0.175 – 0.17 after 14 iterations. Figure 5.10d shows that loss on the test set decreases from 0.15 – 0.09 with iterations.

Comparison of the encoder blocks of the architectures is given in Table 5.6 (Number of parameters in the architectures) and Table 5.7 (Time required per iteration). As fully connected layers of G.A.E. and V.A.E. are replaced by fully convolutional layers, the number of parameters in G.A.E.-F.C. and V.A.E.-F.C. are reduced.

Table 5.6: Number of Parameters in encoder blocks of the auto-encoder architectures

Data-set	**G.A.E.**	**V.A.E.**	**G.A.E.-F.C.**	**V.A.E.-F.C**
Blog-Net	262048	526144	8158	8159
Cora-Net	45888	93824	1403	1404
Cite-Net	118528	239104	3673	3674
Flickr-Net	385504	773056	12016	12017
Protein-Net	1600	2624	19	20
Wiki-Net	1280	2304	9	10

Configuration of the system on which experimental study was conducted is Intel(R) Core(T.M.) i5-6402P CPU@2.8GHz with four cores and 8GB DDR3 RAM. Table 5.7 gives the average time per iteration (in secs) of the auto-encoder architectures

Table 5.7: Average time per iteration (in secs) of the auto-encoder architectures

Data-set	**G.A.E.**	**V.A.E.**	**G.A.E.-F.C.**	**V.A.E.-F.C**
Blog-Net	39 (±1.49)	59 (±1.84)	2.3 (±0.97)	3.5 (±0.67)
Cora-Net	24 (±1.31)	29 (±1.14)	1.23 (±0.64)	1.55 (±0.49)
Cite-Net	30 (±2.04)	37 (±2.11)	1.95 (±0.57)	2.16 (±0.4)
Flickr-Net	63 (±1.36)	89 (±1.91)	4.85 (±0.88)	5.76 (±0.76)
Protein-Net	11 (±0.67)	19 (±1.04)	0.89 (±0.11)	1.06 (±0.19)
Wiki-Net	16 (±0.58)	19 (±0.67)	0.86 (±0.09)	0.96 (±0.04)

The quality of the node embedding learned using the auto-encoder architectures were tested using distance-based statistics. Table 5.8 gives the results of parameter separation index for the data-sets. The L.S.R. obtained using G.A.E.-F.C. were higher (+) or lower (-) on separation index compared to the closest technique as mentioned: Wiki-Net (-0.5%), Cora-Net (-30%), Cite-Net (-0.26%), Blog-Net (-0.15%), Flickr-Net (-17%) and Protein-Net (22%). The L.S.R. obtained using V.A.E.-F.C. were higher (+) or lower (-) on separation index compared to the closest technique as mentioned: Wiki-Net (4%), Cora-Net (-32%), Cite-Net (5%), Blog-Net (23.5%), Flickr-Net (8%) and Protein-Net (38%).

Table 5.8: Comparison of G.A.E., V.A.E., G.A.E.-F.C., V.A.E.-F.C. on separation index

Data-set	**G.A.E.**	**V.A.E.**	**G.A.E.-F.C.**	**V.A.E.-F.C**
Blog-Net	4.24 (±0.76)	43.26 (±3.22)	4.29 (±0.71)	3.26 (±0.54)
Cora-Net	2.29 (±0.37)	37.19 (±3.92)	2.91 (±0.55)	2.93 (±0.57)
Cite-Net	6.18 (±1.16)	60.29 (±4.19)	6.19 (±0.64)	5.76 (±0.3)
Flickr-Net	7.16 (±1.39)	165.27 (±2.82)	8.16 (±0.67)	6.69 (±1.19)
Protein-Net	2.069 (±0.48)	8.16 (±0.81)	1.59 (±0.27)	0.97 (±0.08)
Wiki-Net	3.623 (±0.57)	13.29 (±2.09)	3.67 (±0.27)	2.86 (±0.19)

Table 5.9 gives the results of parameter widest within-cluster gap for the data-sets. The L.S.R. obtained using G.A.E.-F.C. were higher (+) or lower (-) on widest within-cluster gap compared to the closest technique as mentioned: Wiki-Net (11%), Cora-Net (-0.3%), Cite-Net (26%), Blog-Net (20%), Flickr-Net (21%) and Protein-Net (6%). The L.S.R. obtained using V.A.E.-F.C. were higher (+) or lower (-) on widest within-cluster gap compared to the closest technique as mentioned: Wiki-Net (8%), Cora-Net (-2%), Cite-Net (-5%), Blog-Net (55%), Flickr-Net (18%) and Protein-Net (3%).

Table 5.9: Comparison of G.A.E., V.A.E., G.A.E.-F.C., V.A.E.-F.C. on widest within-cluster gap

Data-set	**G.A.E.**	**V.A.E.**	**G.A.E.-F.C.**	**V.A.E.-F.C**
Blog-Net	2.49 (±0.67)	0.929 (±0.41)	3.16 (±0.34)	3.94 (±0.09)
Cora-Net	4.29 (±0.58)	2.549 (±0.73)	4.16 (±0.79)	4.27 (±0.2)
Cite-Net	2.59 (±0.54)	0.765 (±0.29)	2.94 (±0.61)	2.48 (±0.24)
Flickr-Net	4.19 (±0.91)	2.9 (±0.19)	4.93 (±0.52)	4.81 (±0.49)
Protein-Net	3.78 (±0.38)	3.4 (±0.18)	3.9 (±0.24)	3.98 (±0.42)
Wiki-Net	4.065 (±0.83)	3.952 (±0.64)	4.354 (±0.31)	4.289 (±0.67)

Table 5.10 gives the results of the parameter average silhouette width for the data-sets. The L.S.R. obtained using G.A.E.-F.C. were higher (+) or lower (-) on average silhouette width compared to the closest technique as mentioned: Wiki-Net (-4.9%), Cora-Net (2.3%), Cite-Net (-1.2%), Blog-Net (-2%), Flickr-Net (3.21%) and Protein-Net (4.6%). The L.S.R. obtained using V.A.E.-F.C. were higher (+) or lower (-) on average silhouette

width compared to the closest technique as mentioned: Wiki-Net (-2.8%), Cora-Net (2.3%), Cite-Net (-1.5%), Blog-Net (3.5%), Flickr-Net (6%) and Protein-Net (4.8%).

Table 5.10: Comparison of G.A.E., V.A.E., G.A.E.-F.C., V.A.E.-F.C. on average silhouette width

Data-set	**G.A.E.**	**V.A.E.**	**G.A.E.-F.C.**	**V.A.E.-F.C**
Blog-Net	0.19 (±0.0015)	-0.43 (±0.0048)	0.18 (±0.0091)	0.24 (±0.003)
Cora-Net	0.28 (±0.0048)	-0.19 (±0.0041)	0.31 (±0.005)	0.32 (±0.0065)
Cite-Net	0.31 (±0.0029)	-0.18 (±0.0015)	0.29 (±0.0049)	0.27 (±0.0034)
Flickr-Net	0.26 (±0.0048)	-0.64 (±0.0011)	0.34 (±0.007)	0.38 (±0.0019)
Protein-Net	0.37 (±0.0049)	0.16 (±0.0064)	0.43 (±0.0078)	0.44 (±0.0024)
Wiki-Net	0.49 (±0.0058)	0.18 (±0.0081)	0.38 (±0.0064)	0.45 (±0.002)

Table 5.11 gives the results of parameter average distance between clusters for the data-sets. The L.S.R. obtained using G.A.E.-F.C. were higher (+) or lower (-) on average distance between clusters compared to the closest technique as mentioned: Wiki-Net (6.4%), Cora-Net (14%), Cite-Net (-23%), Blog-Net (-6%), Flickr-Net (-23%) and Protein-Net (6.6%). The L.S.R. obtained using V.A.E.-F.C. were higher (+) or lower (-) on average distance between clusters compared to the closest technique as mentioned: Wiki-Net (6.8%), Cora-Net (35%), Cite-Net (-15%), Blog-Net (5%), Flickr-Net (-16%) and Protein-Net (4.8%).

Table 5.11: Comparison of G.A.E., V.A.E., G.A.E.-F.C., V.A.E.-F.C. on average distance between clusters

Data-set	**G.A.E.**	**V.A.E.**	**G.A.E.-F.C.**	**V.A.E.-F.C**
Blog-Net	1.82 (±0.077)	0.49 (±0.059)	1.68 (±0.022)	1.92 (±0.082)
Cora-Net	3.82 (±0.089)	1.23 (±0.081)	4.26 (±0.034)	4.81 (±0.027)
Cite-Net	3.67 (±0.024)	0.96 (±0.088)	3.16 (±0.064)	3.24 (±0.084)
Flickr-Net	2.91 (±0.027)	1.02 (±0.018)	4.24 (±0.046)	3.91 (±0.044)
Protein-Net	1.29 (±0.12)	1.29 (±0.09)	1.84 (±0.18)	1.65 (±0.097)
Wiki-Net	1.21 (±0.091)	1.06 (±0.044)	1.82 (±0.038)	1.86 (±0.049)

Table 5.12 gives the results of parameter Dunn index for the data-sets. The L.S.R.

obtained using G.A.E.-F.C. were higher (+) or lower (-) on Dunn index compared to the closest technique as mentioned: Wiki-Net (-2%), Cora-Net (3%), Cite-Net (-2%), Blog-Net (-4%), Flickr-Net (-2%) and Protein-Net (4.6%). The L.S.R. obtained using V.A.E.-F.C. were higher (+) or lower (-) on Dunn index compared to the closest technique as mentioned: Wiki-Net (1.8%), Cora-Net (2%), Cite-Net (2.15%), Blog-Net (2.5%), Flickr-Net (-6.6%) and Protein-Net (3%).

Table 5.12: Comparison of G.A.E., V.A.E., G.A.E.-F.C., V.A.E.-F.C. on Dunn index

Data-set	**G.A.E.**	**V.A.E.**	**G.A.E.-F.C.**	**V.A.E.-F.C**
Blog-Net	0.23 (±0.019)	0.49 (±0.067)	0.27 (±0.018)	0.20 (±0.002)
Cora-Net	0.43 (±0.051)	0.61 (±0.037)	0.31 (±0.031)	0.37 (±0.014)
Cite-Net	0.19 (±0.046)	0.43 (±0.054)	0.21 (±0.028)	0.16 (±0.008)
Flickr-Net	0.18 (±0.009)	0.51 (±0.015)	0.21 (±0.092)	0.24 (±0.031)
Protein-Net	0.24 (±0.077)	0.26 (±0.034)	0.19 (±0.084)	0.21 (±0.02)
Wiki-Net	0.29 (±0.072)	0.34 (±0.048)	0.31 (±0.025)	0.27 (±0.061)

5.6 Discussion of Results and Summary

Deep learning architectures in the literature are based on Graph Convolutional Networks (G.C.N.). G.C.N. and its variants focus on information aggregation. A drawback of these models is the fully connected layer used in them increases the number of parameters. As the graph size increases, the number of parameters also increases, this leads to slower convergence of gradient descent. Therefore in auto-encoder architectures are proposed which modify existing techniques like Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.) by replacing the fully connected layers with 1-D fully convolutional ones. The modified architectures are referred to as Graph Auto-Encoder with Fully Convolutional layers (G.A.E.-F.C.) and Variational Graph Auto-Encoder with Fully Convolutional layers (V.A.E.-F.C.).

Figure 5.5 showed that the training and test loss on Blog-Net was 0.043 – 0.037 for V.A.E.-F.C. As the parameters of G.A.E. and V.A.E. were 262048 and 264096, respectively, the training time was in the range of 39 – 59 secs compared to that of

the convolutional implementations (2.3 − 3.5 secs). A large number of parameters slowed the optimization. For the current experimental study, the number of epochs was limited to 100. The input size of Blog-Net is $X = 5196 * 8189.A = 5196 * 5196$ would require larger number of epochs on the data. Between the G.A.E.-F.C. and V.A.E.-F.C., the former is a single-layered G.C.N. architecture whereas the later is a two-layered G.C.N. architecture. Consequently, the number of parameters of V.A.E.-F.C (8159) is higher than that of G.A.E.-F.C (8126). During the parameter training, the two-layered architecture saw a decrease in training and test loss with epochs compared to the single-layered architecture. This trend was also observed in Cora-Net ($X = 2708 * 1434, A = 2708 * 2708$), Cite-Net ($X = 3312 * 3704, A = 3312 * 3312$) and Flickr-Net ($X = 7575 * 12047, A = 7575 * 7575$). However, in Protein-Net ($X = 3890 * 50, A = 3890 * 3890$) and Wiki-Net ($X = 4777 * 40, A = 4777 * 4777$) as the input graphs had lesser dimensions, the auto-encoder architectures saw different trends. The convergence required less iterations (G.A.E. = 74, V.A.E. = 36, G.A.E.-F.C. = 14 and V.A.E.-F.C. = 36) and training and test loss was lower (G.A.E. = 0.014, V.A.E. = 4.2, G.A.E.-F.C. = 0.175-0.17 and V.A.E.-F.C. = 0.15-0.09). Table 5.13 gives the results of the transitivity of the auto-encoder architectures on the data-sets. The L.S.R. obtained using G.A.E.-F.C. were higher (+) or lower (-) on transitivity compared to the closest technique as mentioned: Wiki-Net (-10%), Cora-Net (5%), Cite-Net (9%), Blog-Net (3%), Flickr-Net (8%) and Protein-Net (-7%). The L.S.R. obtained using V.A.E.-F.C. were higher (+) or lower (-) on transitivity compared to the closest technique as mentioned: Wiki-Net (-5%), Cora-Net (6%), Cite-Net (7%), Blog-Net (7%), Flickr-Net (3%) and Protein-Net (-2%).

Table 5.13: Comparison of G.A.E., V.A.E., G.A.E.-F.C., V.A.E.-F.C. on transitivity

Data-set	**Actual** c	**G.A.E.**	**V.A.E.**	**G.A.E.-F.C.**	**V.A.E.-F.C**
Blog-Net	0.08	0.19 (±0.032)	0.41 (±0.091)	0.16 (±0.082)	0.11 (±0.032)
Cora-Net	0.09	0.13 (±0.075)	0.56 (±0.064)	0.08 (±0.009)	0.07 (±0.07)
Cite-Net	0.13	0.18 (±0.064)	0.43 (±0.032)	0.09 (±0.064)	0.11 (±0.038)
Flickr-Net	0.1	0.16 (±0.093)	0.34 (±0.082)	0.08 (±0.014)	0.13 (±0.027)
Protein-Net	0.09	0.06 (±0.034)	0.16 (±0.051)	0.13 (±0.028)	0.08 (±0.004)
Wiki-Net	0.43	0.37 (±0.035)	0.47 (±0.064)	0.37 (±0.023)	0.42 (±0.008)

With the advent of the digital transformation of the society, generation of data is occurring at an exponential rate. Network models for representation of data have become popular in scientific domains due to the ease of interpretation as well as analysis. In-spite of such considerable advantages in research, networks or graphs are also associated with disadvantages such as lack of independent observations and the existence of coupling between data-points. Representation learning is proposed as a suitable solution against such drawbacks. It was observed that compared to G.A.E. and V.A.E., the proposed architectures performed better at representation learning. Such architectures were also able to learn representations for data-points not observed in the training phase. This feature makes them more suitable for deployment on real-world systems than heuristic-based shallow encoder techniques. The operation of aggregation and transformation performed in the hidden layer of a neural network is equivalent to laplacian smoothing operation. Hence, Auto-encoder architectures learn representations such that data-points associated with similar characteristics in the high dimension space are located at proximity in the latent space. Experiments performed on network data-sets experimentally validated this theoretical insight.

Chapter 6

INVESTIGATIONS ON RESIDUAL GRAPH CONVOLUTIONAL NETWORK FOR REPRESENTATION LEARNING ON NETWORKS

6.1 Introduction

In Chapter 7.1, two deep auto-encoders frameworks were described - Graph Auto-Encoder with Fully Convolutional layers (G.A.E.-F.C.) and Variational Graph Auto-Encoder with Fully Convolutional layers (V.A.E.-F.C.). Experiments performed on network data-sets demonstrated that for network data-sets with features < 50, standard G.C.N. based auto-encoders such as Graph Auto-Encoder with Fully Convolutional layers (G.A.E.-F.C.) and Variational Graph Auto-Encoder with Fully Convolutional layers (V.A.E.-F.C.) achieved similar test accuracy as Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.). However, as the scale and dimensionality of the data increased the performance of Graph Auto-Encoder (G.A.E.-F.C.) and Variational Graph Auto-Encoder (V.A.E.-F.C.) was better.

The deep graph neural network architectures in the literature as well as proposed in Chapter 7.1 are two layers deep. Therefore, these G.C.N. based architectures have a small receptive field [34, 51]. However, to improve training accuracy and reduce the loss on data, a much deeper architecture would be required. Increasing the depth of these architectures by increasing the hidden layers causes a dramatic drop in model performance. Hence, to build deeper models with many layers of neighborhood aggregation without hampering training time and accuracy, Chapter 7.1 investigates the effects of addition of a residual connection to the Graph Convolutional Network architecture.

The advantages of this approach are demonstrated over other L.S.R. frameworks by performing experiments on network data-sets. The rest of the paper is organized as follows: In Section 6.2 describes the issues in deep learning architectures for L.S.R., Section 6.3 provides a background of Residual Neural Architectures, Section 6.4 describes the performance of the proposed method and discusses the results.

6.2 Issues in Deep Learning Architectures

Theoretically, deep neural networks are accurate, and this accuracy increases with additional hidden layers [34]. However, practically in deep neural networks, the problem of vanishing or exploding gradients is observed. This hampers convergence [36]. As the depth of the networks increases, the number of parameters also increases ($\sim 10^6 - 10^7$), and the accuracy during optimization gets saturated. Adding additional layers in such a scenario affects the training accuracy while increasing the training time. The convolutional implementations in Chapter 7.1 address the issue related to excess parameters. However, to improve the receptive field of the graph convolutional network, there is a need to increase its depth.

6.2.1 Drawbacks of the existing architectures:

- Existing G.C.N. based architectures are two to three layers deep.
- Receptive field of standard G.C.N. based architectures is low.

- G.C.N. based architectures, when stacked together, cause a dramatic drop in model performance [122].

6.2.2 Motivation for the proposed approach

In G.C.N. based networks, a graph convolution operation aggregates node attribute information. In the first convolution layer, only information of directly connected nodes is aggregated. Hence, the receptive field of a standard G.C.N. is less. To increase the receptive field, it is necessary to transfer the activations (learned representations) of the initial convolutions to deeper layers of the network. To achieve this, multiple graph convolution layers could be stacked in a single architecture. However, as the graph convolution is a particular form of Laplacian smoothing, stacked graph convolutions create concerns of over smoothing [32, 35, 124]. To overcome the limits of the G.C.N. model with shallow architectures, the current investigation proposes a Residual Graph Convolutional Network architecture.

6.3 Residual Graph Convolutional Network

6.3.1 Background of residual blocks

For machine learning tasks, it was observed that as the depth of the neural networks was increased, accuracy saturated, and with further training degraded rapidly. This problem was called vanishing or exploding gradients. It was also aggravated with increasing network depth [125]. Residual networks could avoid these problems by utilizing "short-cuts" or skip-connections in the network as shown in Figure 6.1.

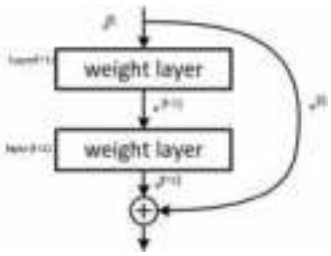


Figure 6.1: A Residual neural network architecture. The skip-connection is used to transfer the activation's to deeper parts of the network.

6.3.2 Architecture

Residual Graph Convolutional Networks architecture is shown in Figure 6.2. Input of layer $l + 1$ is given by $a^{[l]}$, then the two-step activation function computation at layer $l + 1$ is given by $z^{[l+1]} = W^{[l]}a^{[l]} + b^{[l+1]}$ followed by $a^{[l+1]} = g(z^{[l+1]})$. $g(.)$ is a non-linear activation function. Similarly, in layer $l + 2$ the two-step activation function computation is given by $z^{[l+2]} = W^{[l+1]}a^{[l+1]} + b^{[l+2]}$ followed by $a^{[l+2]} = g(z^{[l+2]})$. The change in a residual network is the activation of previous layers $a^{[l]}$ is added to deeper layers, changing $a^{[l+2]}$ to $a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$. To avoid dimension mismatch, the activation of the previous layers $a^{[l]}$ is passed through a dense layer to resize $a^{[l]}$ for addition to $(z^{[l+2]}$. This dense layer does not apply a nonlinear activation function as its role is only to resize the input to match the dimensions of the destination. Using such residual blocks it is possible to build deep networks that avoid problems of over-fitting or vanishing/exploding gradients. During experimentation, batch normalization layer and ReLU activation were found to give optimal results.

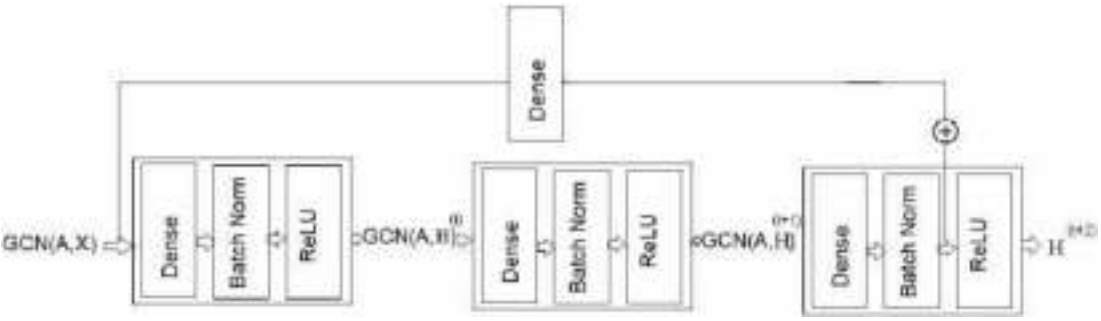


Figure 6.2: Residual graph convolutional network block

6.3.3 Understanding neighborhood-aggregation encoder algorithm of G.C.N. and Residual G.C.N.

Algorithm 7 gives the neighborhood aggregation strategy of a standard G.C.N.. There are two operations: AGGREGATE and COMBINE. The AGGREGATE operation adds the layer k node embedding of all nodes in the neighborhood of node u to u. COMBINE operations applies a nonlinearity to the aggregated node embedding of node u.

Algorithm 7: Neighborhood aggregation algorithm of G.C.N. Adapted from [39]

Inputs: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\{\mathbf{W}^k, \forall k \in [1, K]\}$; non-linearity σ; differentiable aggregator functions $\{\text{AGGREGATE}_k, \forall k \in [1, K]\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Outpt : Vector embedding $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \text{NORMALIZE}(\mathbf{h}_v^k), \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

The Neighborhood aggregation algorithm of a residual G.C.N. is given in Algorithm 8. For every third layer, the AGGREGATE operation adds the layer k and layer $k-2$ node

embedding of all nodes in the neighborhood of node u to u.

Algorithm 8: Neighborhood aggregation algorithm of Residual G.C.N. Adapted from [39]

Inputs: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\{\mathbf{W}^k, \forall k \in [1, K]\}$; non-linearity σ; differentiable aggregator functions $\{\text{AGGREGATE}_k, \forall k \in [1, K]\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Outpt : Vector embedding $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    if $k\%3 == 0$ **then**
      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}, \mathbf{h}_{\mathcal{N}(v)}^{k-2})$;
      $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
    else
      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
      $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
    end
  end
  $\mathbf{h}_v^k \leftarrow \text{NORMALIZE}(\mathbf{h}_v^k), \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

The complexity analysis, for an input graph having N nodes and feature vectors of length C, the adjacency matrix A will be in R^{N*N} and input feature matrix V^a will be of size R^{N*C}. For a given graph convolution layer, to obtain an output $V^{out} = R^{N*F}$, the time complexity is $O(N^2CF)$. For the residual G.C.N. the time complexity is $O(N^2CF + N^2C'F')$ where, component $N'^2C'F'$ is due to the aggregation operation requiring additional computations of the residual block. For the residual block, it is assumed the input graph has N nodes and feature vectors of length C', the adjacency matrix A will be in R^{N*N} and input feature matrix V^a will be of size $R^{N*C'}$.

6.4 Performance of Residual Graph Convolutional Network for Network Representation Learning

The standard Graph Convolutional Network and Residual Graph Convolutional Network architectures were compared in the experimental study. These architectures were used to convert network data of Blog-Net, Cora-Net, Cite-Net, Flickr-Net, Protein-Net and Wiki-Net from their original dimensions into vector space of dimension = 32. The quality of the vertex embeddings is measured using distance-based statistics obtained from clustering the representations. The parameters are separation index, widest within-cluster gap, average silhouette width, the average distance between clusters and Dunn index. Twenty iterations are performed on vertex embeddings for calculating each performance metric. The values of mean along with limits of two standard deviations $2SD$ of the mean (upper and lower bounds of the 95% confidence interval) are given for quantitative comparison. Additionally, the architectures are compared based on their parameters and execution time on the data-sets. T Kipf *et al.* **??** implementation of G.C.N. is used. Residual G.C.N. is built using Keras package in Python [123].

6.4.1 Data-sets

Network data-sets with $|V| > 10^3$ were chosen for the evaluation of Residual G.C.N..

Table 6.1: Description of Network Data-sets with binary attributes

Description	**Blog-Net**	**Flickr-Net**	**Protein-Net**	**Wiki-net**	**Cora-net**	**Cite-net**
$\|V\|$	5196	7575	3890	4777	2708	3312
V^a	8189	12047	50	40	1434	3704
$\|E\|$	171743	239738	37845	54810	5429	4732
c	0.08	0.1	0.09	0.43	0.14	0.13

6.4.2 Architectures

The number of skip-connections used in the Residual Graph Convolutional Network architecture for BlogCatalog, CiteSeer, Cora, Flickr, Protein and Wiki data-sets is given in Table 6.2.

Table 6.2: Number of skip-connections used residual graph convolutional network architectures

Sr No	Data-set	Dense layers	Skip connections
1	Blog-Net	6	2
2	Cora-Net	6	2
3	Cite-Net	5	1
4	Flickr-Net	6	2
5	Protein-Net	3	1
6	Wiki-Net	3	1

Dimensions of the fully connected layers used in the G.C.N. and Residual G.C.N. architectures is summarized in Table 6.3 and Table 6.4 respectively.

Table 6.3: Dimensions of the fully connected layers in the G.C.N. architecture

layer name	Blog-Net	Cora-Net	Cite-Net	Flickr-Net	Protein-Net	Wiki-Net
X	5196*8189	2708*1434	3312*3704	7575*12047	3890*50	4777*40
A	5196*5196	2708*2708	3312*3312	7575*7575	3890*3890	4777*4777
dense_encode	8189*4096	1434*1024	3703*1024	12047*5196	50*128	40*128
dense_encode	4096*1024	1024*512	1024*256	5196 * 2048	128*64	128*64
dense_encode	1024*512	512*256	256*128	2048 * 1024	64*32	64*32
dense_encode	512*128	256*128	128*64	1024 * 512	-	-
dense_encode	128*64	128*64	64*32	512 * 128	-	-
dense_encode	64*32	64*32	-	128 * 32	-	-

Table 6.4: Dimensions of the fully connected layers in the Residual G.C.N. architecture

layer name	Blog-Net	Cora-Net	Cite-Net	Flickr-Net	Protein-Net	Wiki-Net
X	5196*8189	2708*1434	3312*3704	7575*12047	3890*50	4777*40
A	5196*5196	2708*2708	3312*3312	7575*7575	3890*3890	4777*4777
dense_encode	8189*4096	1434*1024	3703*1024	12047*5196	50*128	40*128
dense_encode	4096*1024	1024*512	1024*256	5196 * 2048	128*64	128*64
dense_encode	1024*512	512*256	256*128	2048 * 1024	64*32	64*32
dense_residual	4096*512	1024*256	1024*128	5196*1024	128*32	128*32
dense_encode	512*128	256*128	128*64	1024*512	-	-
dense_encode	128*64	128*64	64*32	512*128	-	-
dense_encode	64*32	64*32	-	128*32	-	-
dense_residual	128*32	128*32	-	512*32	-	-

Table 6.5 gives the number of parameters in the G.C.N. and Residual G.C.N. architectures. A higher number of parameters leads to more computations and also increases the time required for obtaining the L.S.R.

Table 6.5: Number of parameters in the G.C.N. and Residual G.C.N. architectures

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	38336512	40437760
Cora-Net	2166784	2433024
Cite-Net	4096934	4228008
Flickr-Net	75895924	81233012
Protein-Net	16640	20736
Wiki-Net	15360	19456

Configuration of the system on which experimental study was conducted is Intel(R) Core(T.M.) i5-6402P CPU@2.8GHz with four cores and 8GB DDR3 RAM. Table 6.6 gives the time for execution (in secs) of the G.C.N. and Residual G.C.N. architectures.

Table 6.6: Time for execution (in secs) of the G.C.N. and Residual G.C.N. architectures

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	5.4 (±0.36)	7.6 (±0.91)
Cora-Net	7.1 (±0.52)	9.3 (±0.8)
Cite-Net	5 (±0.67)	6.4 (±1.09)
Flickr-Net	11.2 (±0.28)	16.3 (±0.98)
Protein-Net	4.9 (±0.76)	6.2 (±1.03)
Wiki-Net	4.3 (±0.3)	6.7 (±0.28)

6.4.3 Results

The quality of the node embedding generated by the standard Graph Convolutional Network and the Residual Graph Convolutional Network architecture was evaluated using node clustering. The aim was to obtain clusters with high compactness and high separation. Table 6.7 gives the results of parameter separation index for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on separation index compared to the a standard G.C.N. as mentioned: Wiki-Net (8%), Cora-Net (-1%), Cite-Net (-1%), Blog-Net (-11%), Flickr-Net (7%) and Protein-Net (-6.2%).

Table 6.7: Result of separation index of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	**1.43** (±0.061)	1.56 (±0.048)
Cora-Net	**3.15** (±0.083)	3.17 (±0.37)
Cite-Net	**4.29** (±0.49)	4.34 (±0.78)
Flickr-Net	2.84 (±0.028)	**2.68** (±0.41)
Protein-Net	**1.59** (±0.089)	1.67 (±0.078)
Wiki-Net	2.76 (±0.34)	**2.67** (±0.19)

Table 6.8 gives the results of parameter widest within-cluster gap for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on widest within-cluster gap compared to the a standard G.C.N. as mentioned: Wiki-Net (6%), Cora-Net (5%), Cite-Net (-30%), Blog-Net (6%), Flickr-Net (5%) and Protein-Net (-8%).

Table 6.8: Result of widest within-cluster gap of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	2.37 (±0.059)	**2.49** (±0.084)
Cora-Net	4.61 (±0.053)	**4.82** (±0.048)
Cite-Net	**1.64** (±0.06)	1.28 (±0.16)
Flickr-Net	5.83 (±0.57)	**6.08** (±0.6)
Protein-Net	**4.29** (±0.3)	3.97 (±0.46)
Wiki-Net	6.18 (±0.51)	**6.43** (±1.02)

Table 6.9 gives the results of the parameter average silhouette width for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on average silhouette width compared to the a standard G.C.N. as mentioned: Wiki-Net (-4%), Cora-Net (1%), Cite-Net (6%), Blog-Net (7%), Flickr-Net (-4%) and Protein-Net (2%).

Table 6.9: Result of average silhouette width of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	0.21 (±0.0064)	**0.34** (±0.0097)
Cora-Net	0.37 (±0.0048)	**0.39** (±0.0094)
Cite-Net	0.16 (±0.002)	**0.22** (±0.0088)
Flickr-Net	**0.43** (±0.0109)	0.38 (±0.0029)
Protein-Net	0.35 (±0.0043)	**0.38** (±0.0097)
Wiki-Net	**0.57** (±0.0046)	0.53 (±0.0046)

Table 6.10 gives the results of parameter average distance between clusters for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on average distance between clusters compared to the a standard G.C.N. as mentioned: Wiki-Net (5%), Cora-Net (5%), Cite-Net (-68%), Blog-Net (3%), Flickr-Net (-11%) and Protein-Net (2%).

Table 6.10: Result of average distance between clusters of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	0.57 (±0.0028)	**0.62** (±0.0064)
Cora-Net	1.58 (±0.003)	**1.65** (±0.0074)
Cite-Net	**0.38** (±0.0091)	0.24 (±0.0027)
Flickr-Net	**3.16** (±0.0206)	2.94 (±0.071)
Protein-Net	3.46 (±0.26)	**3.51** (±0.4)
Wiki-Net	1.67 (±0.156)	**1.93** (±0.074)

Table 6.11 gives the results of parameter Dunn index for the data-sets. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on Dunn index compared to the a standard G.C.N. as mentioned: Wiki-Net (-5%), Cora-Net (-5%), Cite-Net (-12%), Blog-Net (11%), Flickr-Net (4%) and Protein-Net (0.12%).

Table 6.11: Result of Dunn index of G.C.N. and Residual G.C.N.

Data-set	**G.C.N.**	**Residual G.C.N.**
Blog-Net	0.87 (±0.0059)	**0.73** (±0.0027)
Cora-Net	**0.63** (±0.0049)	0.68 (±0.0067)
Cite-Net	**0.81** (±0.0042)	0.94 (±0.004)
Flickr-Net	0.34 (±0.0087)	**0.26** (±0.0091)
Protein-Net	0.29 (±0.0082)	0.29 (±0.0014)
Wiki-Net	**0.44** (±0.0037)	0.46 (±0.0084)

6.4.4 Discussion of results and summary

Graph Convolutional Networks (G.C.N.) are capable of encoding both graph structure and node features in the vector representations. The standard G.C.N. and its variants are two layers deep. Hence, to overcome the limitations of shallow architectures, the current investigation proposes a Residual Graph Convolutional Network architecture. Adding a residual block to plain neural networks had proven useful in increasing their depth without affecting the performance. However, the addition of a residual block to a G.C.N. to improve its receptive field was not observed in the literature. In the current

experimental setting, it was noted that Residual G.C.N. had comparable results with the standard G.C.N.

For the Blog-Net data-set, the performance of Residual G.C.N. was lower by 11% on the separation index, higher by 6% on the widest within-cluster gap, higher by 7% on average silhouette width, higher by 3% on the average distance between clusters and higher by 11% on Dunn index. For the Cora-Net data-set, the performance of Residual G.C.N. was lower by 1% on the separation index, higher by 5% on the widest within-cluster gap, higher by 1% on average silhouette width, higher by 5% on the average distance between clusters and lower by 5% on Dunn index. For the Cite-Net data-set, the performance of Residual G.C.N. was lower by 1% on separation index, lower by 30% on the widest within-cluster gap, higher by 6% on average silhouette width, lower by 68% on the average distance between clusters and lower by 12% on Dunn index. For the Flickr-Net data-set, the performance of Residual G.C.N. was higher by 7% on the separation index, higher by 5% on the widest within-cluster gap, lower by 4% on average silhouette width, lower by 11% on the average distance between clusters and higher by 4% on Dunn index. For the Protein-Net data-set, the performance of Residual G.C.N. was lower by 6.2% on separation index, lower by 8% on the widest within-cluster gap, higher by 2% on average silhouette width, higher by 2% on the average distance between clusters and higher by 0.12% on Dunn index. For the Wiki-Net data-set, the performance of Residual G.C.N. was higher by 8% on separation index, higher by 6% on the widest within-cluster gap, lower by 2% on average silhouette width, higher by 5% on the average distance between clusters and lower by 5% on Dunn index.

Table 6.12 gives the result of the transitivity of G.C.N. and Residual G.C.N. The L.S.R. obtained using Residual G.C.N. were higher (+) or lower (-) on transitivity compared to the G.C.N. as mentioned: Wiki-Net (12%), Cora-Net (35%), Cite-Net (39%), Blog-Net (38%), Flickr-Net (42%) and Protein-Net (27%).

Table 6.12: Result of transitivity of G.C.N. and Residual G.C.N.

Data-set	**Actual** c	**G.C.N.**	**Residual G.C.N.**
Blog-Net	0.08	0.26 (±0.064)	0.17 (±0.019)
Cora-Net	0.09	0.19 (±0.027)	0.11 (±0.02)
Cite-Net	0.13	0.24 (±0.032)	0.16 (±0.087)
Flickr-Net	0.1	0.22 (±0.096)	0.13 (±0.076)
Protein-Net	0.09	0.19 (±0.103)	0.14 (±0.084)
Wiki-Net	0.43	0.64 (±0.036)	0.57 (±0.094)

Graph convolution is a particular form of Laplacian smoothing. Multiple stacked graph convolutions in the standard G.C.N. lead to over smoothing of the vector representation. This causes the vector representations of the nodes in the original graph are mapped to close to each other in the embedded space. Hence, the reconstructed graph from a standard G.C.N. has higher transitivity than seen in the original graph. However, with the residual blocks, the representations of the initial layers are added to the deeper layers. This addition overcomes, to a certain extent, the smoothing effect of the graph convolution. Therefore, the transitivity of the reconstructed graph is lower in the Residual G.C.N. compared to the standard G.C.N.

Chapter 7

CONCLUSION AND FUTURE SCOPE

The thesis investigated extending network analytical techniques to process non-euclidean data, such as networks (graphs). This domain had received comparatively low levels of attention. There was no straight forward method to analyze network data as learning models were designed for simple euclidean data or grids. Additionally, graph-structured data had issues related to the lack of independent data-points, computationally costly calculations for graph statistics, in-applicability of parallel or distributed algorithms and the curse of dimensionality.

To leverage the potential of network data, there was a need for latent space representation learning techniques that could accommodate high dimensionality, heterogeneity, etc. present in network data in the representations. These representations could then be used to improve the efficiency and performance of downstream network-based applications such as node clustering, node classification, etc.

To address such challenges, the focus of this dissertation was on Latent Space Representations (L.S.R.) of networks, i.e., learning low dimension vector representations of network data. L.S.R. techniques have a data-driven mechanism for learning vector embedding of network data structures. The objectives are mentioned below:

- Survey state of the art Latent Space Representation (L.S.R.) techniques in the

literature.

- Perform extensive experiments on network data-sets from the Stanford Network Analysis Project (SNAP) for understanding the structure and behavior of systems.
- Develop ensemble methods for L.S.R. and perform experiments on network data to compare their performance vis-a-vis existing techniques.
- Propose a convolutional implementation of G.C.N. auto-encoder architectures for L.S.R. to resolve the issues of standard graph auto-encoder architectures such as Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.).
- Increasing the depth of the graph convolutional network architecture for L.S.R. by use of residual blocks to address the problem of a small receptive field of G.C.N.

7.1 Thesis Summary with Overall Conclusion

In Chapter 2, a survey of state of the art Latent Space Representation (L.S.R.) techniques in the literature were presented. In the survey, three categories of grouping, Latent space representation techniques for networks were examined in detail. These were Probabilistic models, Statistical models, and Network representation learning models. Probabilistic models had limited flexibility and hence resulted in poor fits to data. Even though probabilistic models were "generative," they did not generate networks that shared many properties with the specific network they were fit to. Statistical models such as S.B.M. and latent variable models were feasible for data-sets with nodes in the range of 10^{1-3}. This was due to the computationally costly stochastic calculations in such models. The three classes of N.R.L. techniques discussed in the literature were Adjacency preserving methods [24, 25, 26, 30, 31, 84, 85, 86, 87, 88], Multi-hop distance preserving methods and Random walk occurrence preserving methods [89, 90, 91, 92, 93, 94, 95, 96, 97, 98]. The methods of these three classes relied on the concept of "similarity," making them sensitive to the type of data. Apart from these, deep learning based N.R.L. techniques were surveyed in the literature. A drawback of deep learning architectures was the complexity involved in building deep graph

neural network models. The backbone of deep graph neural networks was the graph convolution operation. Graph convolution relied on a fully connected layer. This layer increased the number of parameters of the model and slowed the gradient descent process. It was also observed that G.C.N. based architecture was shallow (two layers) which affected their receptive field. Thus, the survey provided an overview of research gaps in the latent space representation techniques. It also gave a technical foundation for the remaining parts of this dissertation.

In Chapter 3, Network science was applied to understand the trends in domains of scientific literature such as social networking websites (Twitter, Google+), blogs (Blog.com), photo-sharing websites (Flickr), protein-protein interactions (Protein-Net), citation networks (CORA and CiteSeer), transportation networks (High-Net), sexual contact network (Grey-Net), trade network (Trade-Net) and bill co-sponsorship network (Bill-Net). Trends identified using network analysis were the rate of information diffusion, the dominance of certain entities, the number of social contacts and community structure. These trends would not be easily identifiable using relational databases. Although networks are present in every domain, their analysis revealed that the structural characteristics shared by them are similar, i.e., low average path, low diameter, etc. It was also noted that networks capture the behavior of the entities present in them. Using the concepts of graph theory, it was possible to make a statistically valid analysis of these systems and provide insights into their growth. Networks across different domains saw low edge density and the presence of inequality. Networks such as Twt-Net, Gplus-Net, Flickr-Net, Wiki-Net, Blog-Net, Grey-Net and Bill-Net are a particular type of network called "social networks". Social networks represent the sum of all professional, friendship, or family ties of the actors involved in them. Social networks were observed to have higher edge density and average degree compared to other networks. They also had high transitivity, low diameter, and negative assortativity. In the experimental setting, it was found that the two L.S.R. models for conducting the analysis (Stochastic Block model and Latent variable model) had high computational complexity, ignored attribute information and were only suitable for data-sets with nodes in the range of $10^1 - 3$. Therefore, it was necessary to investigate models that could scale to more extensive networks (10^3+).

In Chapter 4, N.R.L. methods were used for network analysis as the networks in this

chapter were large, with nodes in the range of 10^3. The frameworks in the literature were based on the optimization of a single objective function. This could lead to node embedding that is favorable for specific applications but unsuitable for others [48, 49, 50]. It was speculated that a single model was unlikely to capture the entire underlying structure of the data. Hence, to achieve optimal representations, integrating multiple models might improve the accuracy of representation learning significantly. This gap served as the principal motivation for this study. Two ensemble models were proposed viz. Weighted-Average and Mean, in which representation learning of base models (Adjacency preserving methods, Multi-hop distance preserving methods and Random walk occurrence preserving methods) were combined by consensus. In the experimental setting, it was seen that in networks where the Gini index and transitivity were low, all three base models had near equal weight. In networks, that had a high Gini index, adjacency preserving model had higher weightage in the ensemble. The networks with high transitivity gave more weightage to Multi-hop distance preserving model, and networks with high diameter gave more weightage to Random walk occurrence preserving models. A key drawback of ensemble techniques was the computation complexity. The ensemble techniques have a computation complexity of $O(|E|d) + O(|V|d) + O(|E|d)$. This is a summation of the computation complexities of the base models. Constant time is required for performing the combination by consensus operation, i.e., weighted-average or mean. The ensemble N.R.L. methods also ignored the attribute information associated with the networks. Hence, it was necessary to propose models that could consider attribute information associated with the nodes.

In Chapter , Graph Convolutional Network based models were investigated. These models considered both topographical and attribute information in the vector representations. However, the drawback of the G.C.N. was the presence of a fully connected layer. The fully connected layer led to an increase in parameters as the input graph size increased. This led to a slow convergence of gradient descent and increased the time per iteration. Hence, the inquiry proposed to modify the existing Graph Auto-Encoder (G.A.E.) and Variational Graph Auto-Encoder (V.A.E.) by replacing the fully connected layers with 1-D convolutional ones. The proposed models were named as Graph Auto-Encoder with Convolutional layers (G.A.E.-F.C.) and Variational Graph Auto-Encoder with Convolutional layers (V.A.E.-F.C.). Experiments were performed

to compare G.A.E. and V.A.E. with G.A.E.-F.C. and V.A.E.-F.C. The parameters for comparison were the parameters of the architectures, average time per iteration (in secs), transitivity and clustering quality measures such as Separation index, Widest within-cluster gap, Average silhouette width, Average distance between clusters and Dunn index.

In the experimental setting, the performance of G.A.E.-F.C. compared to G.A.E. and V.A.E. for the parameter transitivity showed an improvement of $3-9\%$ on Cora-Net, Cite-Net, Blog-Net and Flickr-Net. Whereas it was lower by $7-10\%$ on Wiki-Net and Protein-Net. Similarly, in V.A.E.-F.C. it showed an improvement of $3-7\%$ on Cora-Net, Cite-Net, Blog-Net and Flickr-Net but was lower by $2-5\%$ on Protein and Wiki-Net. The performance of G.A.E.-F.C. compared to G.A.E. and V.A.E. for the parameter Dunn index showed an improvement of $3-4.6\%$ on Cora-Net and Protein-Net but was lower by $2-4\%$ on Wiki-Net, Cite-Net, Blog-Net and Flickr-Net. Similarly, in V.A.E.-F.C. it showed an improvement of $1.8-3\%$ on Wiki-Net, Cora-Net, Cite-Net, Blog-Net and Protein-Net but was lower by 6.6% on Flickr-Net. The performance of G.A.E.-F.C. compared to G.A.E. and V.A.E. for the parameter average distance between clusters showed an improvement of $6.4-14\%$ on Wiki-Net, Cora-Net and Protein-Net but was lower by $6-23\%$ on Cite-Net, Blog-Net and Flickr-Net. Similarly, in V.A.E.-F.C. it showed an improvement of $4.8-35\%$ on Wiki-Net, Cora-Net, Blog-Net and Protein-Net but was lower by $15-16\%$ on Cite-Net and Flickr-Net. The performance of G.A.E.-F.C. compared to G.A.E. and V.A.E. for the parameter average silhouette width was improved by $2.3-4.6\%$ on Cora-Net, Flickr-Net and Protein-Net but was lower by $1.2-4.9\%$ on Wiki-Net, Cite-Net and Blog-Net. Similarly, in V.A.E.-F.C. it showed an improvement of $2.3-6\%$ on Cora-Net, Blog-Net, Flickr-Net and Protein-Net but was lower by $1.5-2.8\%$ on Wiki-Net and Cite-Net. The performance of G.A.E.-F.C. compared to G.A.E. and V.A.E. for the parameter widest within-cluster gap showed an improvement of $6-26\%$ on Wiki-Net, Cite-Net, Blog-Net, Flickr-Net and Protein but was lower by 0.3% on Cora-Net. Similarly, in V.A.E.-F.C. it showed an improvement of $3-55\%$ on Wiki-Net, Blog-Net, Flickr-Net and Protein-Net but lower by $2-5\%$ on Cora-Net and Cite-Net. The performance of G.A.E.-F.C. compared to G.A.E. and V.A.E. for the parameter separation index showed an improvement of 22% on Protein-Net but was lower by $0.15-30\%$ on Wiki-Net, Cora-Net, Cite-Net, Blog-Net

and Flickr-Net. Similarly, in V.A.E.-F.C. it showed an improvement of 4 – 38% on Wiki-Net, Cite-Net, Blog-Net, Flickr-Net and Protein-Net but was lower by 32% on Cora-Net. The number of parameters in the convolutional implementations was $\frac{1}{100} - \frac{1}{35}$ that of G.A.E. and V.A.E. The average time per iteration (in secs) was $\frac{1}{156} - \frac{1}{14}$ that of G.A.E. and V.A.E. Both G.A.E.-F.C. and V.A.E.-F.C. were "shallow" architectures with two layers deep and therefore had small receptive fields.

In Chapter , an attempt was made to address the issue of shallow G.C.N. architectures by the inclusion of a residual block to increase their depth. Experiments were performed to compare a standard G.C.N. with a G.C.N. having residual blocks. The parameters for comparison were the parameters of the architectures, execution time (in secs), transitivity and clustering quality measures such as Separation index, Widest within-cluster gap, Average silhouette width, Average distance between clusters and Dunn index.

In the experimental setting, the performance of Residual G.C.N. compared to a standard G.C.N. for the parameter transitivity showed an improvement of 12 – 42% on the data-sets. For the parameter Dunn index, the performance of Residual G.C.N. compared to a standard G.C.N. was improved by 0.12 – 11% on Protein-Net, Flickr-Net and Blog-Net but was lower by 5 – 12% on Wiki-Net, Cora-Net and Cite-Net. For the parameter average distance between clusters, the performance of Residual G.C.N. compared to a standard G.C.N. showed an improvement of 2 – 5% on Wiki-Net, Cora-Net, Blog-Net and Protein-Net but was lower by 11 – 68% on Cite-Net and Flickr-Net. For the parameter average silhouette width, the performance of Residual G.C.N. compared to a standard G.C.N. showed an improvement of 1 – 7% on Cora-Net, Cite-Net and Blog-Net but was lower by 4% on Wiki-Net and Flickr-Net. For the parameter widest within-cluster gap, performance of Residual G.C.N. compared to a standard G.C.N. showed an improvement of 5 – 6% on Wiki-Net, Cora-Net, Blog-Net and Flickr-Net but was lower by 8 – 30% on Cite-Net and Protein-Net. For the parameter separation index, performance of Residual G.C.N. compared to a standard G.C.N. showed an improvement of 7 – 8% on Wiki-Net and Flickr-Net but was lower by 1 – 11% on Cora-Net, Cite-Net, Blog-Net and Protein-Net. The time for execution of Residual G.C.N. compared to a standard G.C.N. was higher by 22 – 55% on the data-sets as the

parameters were more by 6 – 43%.

7.2 Future Scope

This dissertation explored several vital directions for network representation learning, but many questions remain unanswered, and many future directions remain open to exploring.

7.2.1 Receptive field

The receptive field of a node refers to the region in the input space that a particular node is affected by. As graphs do not have fixed structures like grids, the neighborhood of nodes differs. The receptive field of the auto-encoder and ensemble models proposed in this investigation can be made more representative using sampling strategies.

7.2.2 Scalability

The proposed work does not scale to graphs with millions of nodes. The main reason is excessive parameters if multiple hidden layers are used, leading to the high complexity of backpropagation. The efficiency of the model can be improved using sub-graph training techniques.

7.2.3 Dynamics

Models proposed in this thesis focus on static graphs i.e., graph structures are assumed to be fixed. However, these models need to be extended to scenarios where network topology changes at any time.

Publications

International Journals

1. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "Understanding attribute and social circle correlation in social networks"; *Turkish Journal of Electrical Engineering & Computer Sciences*, 27(2), pp.1228-1242.

2. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "Modeling influence on a Social Network using Interaction Characteristics"; *International Journal of Computer & Mathematical Sciences*, 6(8), pp.152-160.

3. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "Measurement of network based and random meetings in social networks"; *Turkish Journal of Electrical Engineering & Computer Sciences*, 27(2), pp.765-779.

4. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "Exploring convolutional auto-encoders for representation learning on networks"; *Computer Science Journal*, Accepted.

5. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "Empirical Analysis of Synthetic and Real Networks"; *International Journal of Information Technology*, Accepted. Springer.

6. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "Understanding structure and behavior of systems: A network perspective"; *International Journal of Information Technology*, Accepted. Springer.

National Journals

1. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "Investigations on Residual Auto-encoders for unsupervised network representation learning"; *Journal of The Institution of Engineers (India): Series B*, Communicated. Springer.

International Conferences

1. **Nerurkar, P.**, Shirke, A., Chandane, M. and Bhirud, S., 2018. "Empirical Analysis of Data Clustering Algorithms"; *Procedia Computer Science*, 125, (pp. 770-779). Elsevier.

2. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "A Comparative Analysis of Community Detection Algorithms on Social Networks"; In *Computational Intelligence: Theories, Applications and Future Directions*-Volume I (pp. 287-298). Springer.

3. **Nerurkar, P.**, Shirke, A., Chandane, M. and Bhirud, S., 2018. "A novel heuristic for evolutionary clustering"; *Procedia Computer Science*, 125, (pp. 780-789). Elsevier.

4. **Nerurkar, P.**, Chandane, M., and Bhirud, S., 2019. "Representation learning for social networks using Homophily based Latent Space Model"; In *Proceedings of the International Conference on Omni-Layer Intelligent Systems* (pp. 38-43). ACM.

5. **Nerurkar, P.**, Chandane, M. and Bhirud, S., 2019. "Community Detection Using Node Attributes: A Non-negative Matrix Factorization Approach"; In *Computational Intelligence: Theories, Applications and Future Directions-Volume I* (pp. 275-285). Springer.

Bibliography

[1] Albert-László Barabási et al. *Network science*. Cambridge university press, 2016. xiv, 1, 2, 3, 20

[2] Denny M. Social network analysis. *Institute for Social Science Research, University of Massachusetts, Amherst*, 2014. 1, 9, 24

[3] Denny M. Intermediate social network theory. *Institute for Social Science Research, University of Massachusetts, Amherst*, 2015. 1

[4] Matthew O Jackson. *Social and economic networks*. Princeton university press, 2010. 1, 17, 18

[5] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems*, pages 6530–6539, 2017. 1

[6] Henrique F de Arruda, Filipi N Silva, Cesar H Comin, Diego R Amancio, and Luciano da F Costa. Connecting network science and information theory. *Physica A: Statistical Mechanics and its Applications*, 515:641–648, 2019. 1

[7] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018. 1

[8] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016. 1

[9] Leonardo FS Scabini, Rayner HM Condori, Wesley N Gonçalves, and Odemir M Bruno. Multilayer complex network descriptors for color-texture characterization. *Information Sciences*, 2019. 1

[10] Pranav Nerurkar, Madhav Chandane, and Sunil Bhirud. A comparative analysis of community detection algorithms on social networks. In *Computational Intelligence: Theories, Applications and Future Directions-Volume I*, pages 287–298. Springer, 2019. 1

[11] Pranav Nerurkar, Archana Shirke, Madhav Chandane, and Sunil Bhirud. A novel heuristic for evolutionary clustering. *Procedia Computer Science*, 125:780–789, 2018. 1

[12] Haochen Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. A tutorial on network embeddings. *arXiv preprint arXiv:1808.02590*, 2018. 1

[13] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*, 2019. 1

[14] István A Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, et al. Network-based prediction of protein interactions. *Nature communications*, 10(1):1240, 2019. 1

[15] Roberto Interdonato, Martin Atzmueller, Sabrina Gaito, Rushed Kanawati, Christine Largeron, and Alessandra Sala. Feature-rich networks: going beyond complex network topologies. *Applied Network Science*, 4(1):4, 2019. 1

[16] Igor Linkov and Benjamin D Trump. Applications of network science and systems thinking. In *The Science and Practice of Resilience*, pages 167–179. Springer, 2019. 1

[17] Vicenc Gomez, Andreas Kaltenbrunner, and Vicente Lopez. Statistical analysis of the social network and discussion threads in slashdot. In *Proceedings of the 17th international conference on World Wide Web*, page 645—654. ACM, 2008. 2, 3

[18] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016. xiv, 2, 3, 4, 5, 6, 7, 19

[19] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. 2

[20] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998. 2

[21] Mary McGlohon, Leman Akoglu, and Christos Faloutsos. Statistical properties of social networks. In *Social network data analytics*, pages 17–42. Springer, 2011. 3

[22] Shi Y, Gui H, Zhu Q, Kaplan L, and Han J. Aspem: Embedding learning by aspects in heterogeneous information networks. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 144–152. SIAM, 2018. 4, 8, 31, 36

[23] Perozzi B, Kulkarni V, Chen H, and Skiena S. Don't walk, skip!: Online learning of multi-scale network embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 258–265. ACM, 2017. 8, 33, 36

[24] Tang J, Qu M, Wang M, Zhang M, Yan J, and Mei Q. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015. 8, 25, 30, 36, 77, 83, 128

[25] Huang X, Li J, and Hu X. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739. ACM, 2017. 8, 25, 32, 36, 77, 128

[26] Huang X, Li J, and Hu X. Accelerated attributed network embedding. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 633–641. SIAM, 2017. 8, 25, 32, 36, 77, 128

[27] Cui P, Wang X, Pei J, and Zhu W. A survey on network embedding. *IEEE T Knowl Data En*, 2018. 8, 9, 83

[28] Zitnik M, Agrawal M, and Leskovec J. Modeling polypharmacy side effects with graph convolutional networks. *arXiv preprint arXiv:1802.00543*, 2018. 8, 25, 38

[29] Zitnik M and Leskovec J. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33:190–198, 2017. 8, 25, 38

[30] Liao L, He X, Zhang H, and Chua TS. Attributed social network embedding. *arXiv preprint arXiv:1705.04969*, 2017. 8, 25, 77, 128

[31] Bandyopadhyay S, Kara H, Kannan A, and Murty MN. Fscnmf: Fusing structure and content via non-negative matrix factorization for embedding information networks. *arXiv preprint arXiv:1804.05313*, 2018. 8, 25, 29, 36, 77, 79, 128

[32] Hamilton W, Ying Z, and Leskovec J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017. 9, 35, 37, 45, 115

[33] Butts CT Goodreau SM Pavel NK Bender-deMoll S Morris M Handcock MS, Hunter DR. *statnet: Software Tools for the Statistical Analysis of Network Data*. The Statnet Project, 2016. 9, 23

[34] Zhang M, Cui Z, Neumann M, and Chen Y. An end-to-end deep learning architecture for graph classification. In *Proceedings of AAAI Conference on Artificial Inteligence*, 2018. 10, 11, 25, 36, 43, 91, 114

[35] Kipf TN and Welling M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 10, 25, 36, 37, 91, 92, 115

[36] Chen J, Ma T, and Xiao C. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018. 10, 43, 91, 114

[37] Wu L, Fisch A, Chopra S, Adams K, Bordes A, and Weston J. Starspace: Embed all the things! *arXiv preprint arXiv:1709.03856*, 2017. 10, 91

[38] Donnat C, Zitnik M, Hallac D, and Leskovec J. Spectral graph wavelets for structural role similarity in networks. *arXiv preprint arXiv:1710.10321*, 2017. 10, 91

[39] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017. 10, 36, 45, 96, 117, 118

[40] Goodreau SM. Advances in exponential random graph (p*) models applied to a large social network. *Soc networks*, 29:231–248, 2007. 11, 43, 45, 46

[41] Hoff PD, Raftery AE, and Handcock MS. Latent space approaches to social network analysis. *J Am Stat Assoc*, 97:1090–1098, 2002. 11, 45

[42] Yun Zhang, Kehui Chen, Allan Sampson, Kai Hwang, and Beatriz Luna. Node features adjusted stochastic block model. *Journal of Computational and Graphical Statistics*, pages 1–22, 2019. 11, 21, 45

[43] Soumyasundar Pal and Mark Coates. Scalable mcmc in degree corrected stochastic block model. In *Proc. Intl. Conf. Acoust., Speech and Signal Proc*, 2019. 11, 45

[44] Xiaoyan Lu and Boleslaw K Szymanski. Regularized stochastic block model for robust community detection in complex networks. *arXiv preprint arXiv:1903.11751*, 2019. 11, 46

[45] Marina S Paez, Arash A Amini, and Lizhen Lin. Hierarchical stochastic block model for community detection in multiplex networks. *arXiv preprint arXiv:1904.05330*, 2019. 11, 46

[46] Hannu Reittu, Ilkka Norros, Tomi Räty, Marianna Bolla, and Fülöp Bazsó. Regular decomposition of large graphs: Foundation of a sampling approach to stochastic block model fitting. *Data Science and Engineering*, pages 1–17, 2019. 11, 46

[47] Marianna Pensky, Teng Zhang, et al. Spectral clustering in the dynamic stochastic block model. *Electronic Journal of Statistics*, 13(1):678–709, 2019. 11, 46

[48] Balasubramanian M and Schwartz EL. The isomap algorithm and topological stability. *Science*, 295:7–7, 2002. 11, 29, 36, 77, 130

[49] Roweis ST and Saul LK. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000. 11, 29, 36, 77, 130

[50] Goyal P and Ferrara E. Graph embedding techniques, applications, and performance: A survey. *Knowl-Based Syst*, 151:78–94, 2018. 11, 77, 130

[51] Ryan A Rossi, Rong Zhou, and Nesreen K Ahmed. Deep inductive network representation learning. In *Companion of the The Web Conference 2018 on The Web Conference 2018*, pages 953–960. International World Wide Web Conferences Steering Committee, 2018. 11, 114

[52] Saket Gurukar, Priyesh Vijayan, Aakash Srinivasan, Goonmeet Bajaj, Chen Cai, Moniba Keymanesh, Saravana Kumar, Pranav Maneriker, Anasua Mitra, Vedang Patel, et al. Network representation learning: Consolidation and renewed bearing. *arXiv preprint arXiv:1905.00987*, 2019. 16

[53] Eitan Muller and Renana Peres. The effect of social networks structure on innovation performance: A review and directions for research. *International Journal of Research in Marketing*, 36(1):3–19, 2019. 16

[54] Hao Yin, Austin R Benson, and Jure Leskovec. The local closure coefficient: A new perspective on network clustering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 303–311. ACM, 2019. 16

[55] Tiandong Wang and Sidney I Resnick. Consistency of hill estimators in a linear preferential attachment model. *Extremes*, 22(1):1–28, 2019. 17

[56] Carter T Butts. A dynamic process interpretation of the sparse ergm reference model. *The Journal of Mathematical Sociology*, 43(1):40–57, 2019. 17

[57] Cristian Candia, C Jara-Figueroa, Carlos Rodriguez-Sickert, Albert-László Barabási, and César A Hidalgo. The universal decay of collective memory and attention. *Nature Human Behaviour*, 3(1):82, 2019. 17

[58] Adam Sealfon and Jonathan Ullman. Efficiently estimating erdos-renyi graphs with node differential privacy. *arXiv preprint arXiv:1905.10477*, 2019. 18

[59] Matthew O Jackson and Brian W Rogers. Meeting strangers and friends of friends: How random are social networks? *American Economic Review*, 97(3):890–915, 2007. 19

[60] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 19

[61] Ba-Dung Le, Hong Shen, Hung Nguyen, and Nickolas Falkner. Improved network community detection using meta-heuristic based label propagation. *Applied Intelligence*, 49(4):1451–1466, 2019. 19

[62] Zheng-Hong Deng, Hong-Hai Qiao, Ming-Yu Gao, Qun Song, and Li Gao. Complex network community detection method by improved density peaks model. *Physica A: Statistical Mechanics and its Applications*, 526:121070, 2019. 19

[63] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):118–129, 2009. 19

[64] Clara Granell, Richard K Darst, Alex Arenas, Santo Fortunato, and Sergio Gómez. Benchmark model to assess community structure in evolving networks. *Physical Review E*, 92(1):12–19, 2015. 19

[65] Pattison P Wasserman S. Logit models and logistic regressions for social networks: I. an introduction to markov graphs and p. *Psychometrika*, 61(3):401–425, 1996. 21

[66] MEJ Park J, Newman. Solution for the properties of a clustered network. *Phys Rev E*, 72(2):026136, 2005. 21

[67] Prusaczyk B, Maki J, Luke DA, and Lobb R. Rural health networks: How network analysis can inform patient care and organizational collaboration in a rural breast cancer screening network. *J Rural Health*, 2018. 22

[68] Mônica da Silva Santana, Everardo Valadares de Sá Barretto Sampaio, Vanderlise Giongo, Rômulo Simões Cezar Menezes, Kennedy Nascimento de Jesus, Eliza Rosário Gomes Marinho de Albuquerque, Diego Marcelino do Nascimento, Frans Germain Corneel Pareyn, Tony Jarbas Ferreira Cunha, Raquel Menezes Bezerra Sampaio, et al. Carbon and nitrogen stocks of soils under different land uses in pernambuco state, brazil. *Geoderma Regional*, 16:e00205, 2019. 22

[69] Diaconis P Chatterjee S. Estimating and understanding exponential random graph models. *Ann Stat*, 41(5):2428–2461, 2013. 22, 76

[70] Kalish Y Lusher D Robins G, Pattison P. An introduction to exponential random graph (p*) models for social networks. *Soc networks*, 29(2):173–191, 2007. 22

[71] Tom AB Snijders. Markov chain monte carlo estimation of exponential random graph models. *J Soc Struct*, 3(2):1–40, 2002. 23

[72] Snijders T Moody J Besag J Handcock MS, Robins G. Assessing degeneracy in statistical models of social networks. In *J Am Stat Assoc*. Citeseer, 2003. 23, 76

[73] George G Vega Yon and Kayla de la Haye. Exponential random graph models for little networks. *arXiv preprint arXiv:1904.10406*, 2019. 23

[74] Sly A Bhamidi S, Bresler G. Mixing time of exponential random graphs. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 803–812. IEEE, 2008. 23

[75] Butts CT Goodreau SM Morris M Handcock MS, Hunter DR. statnet: Software tools for the representation, visualization, analysis and simulation of network data. *J Stat Softw*, 24:1–11, 2008. 23

[76] Tjeerd Zandberg and Mark Huisman. Missing behavior data in longitudinal network studies: the impact of treatment methods on estimated effect parameters in stochastic actor oriented models. *Social Network Analysis and Mining*, 9(1):8, 2019. 23

[77] Beth Prusaczyk, Julia Maki, Douglas A Luke, and Rebecca Lobb. Rural health networks: How network analysis can inform patient care and organizational collaboration in a rural breast cancer screening network. *The Journal of Rural Health*, 35(2):222–228, 2019. 23

[78] Pranav Nerurkar, Madhav Chandane, and Sunil Bhirud. Measurement of network based and random meetings in social networks. *Turkish Journal Of Electrical Engineering & Computer Sciences*, 27(2):765–779, 2019. 23

[79] Pranav Nerurkar, Madhav Chandane, and Sunil Bhirud. Understanding attribute and social circle correlation in social networks. *Turkish Journal Of Electrical Engineering & Computer Sciences*, 27(2):1228–1242, 2019. 23

[80] Kenji Yamanishi, Tianyi Wu, Shinya Sugawara, and Makoto Okada. The decomposed normalized maximum likelihood code-length criterion for selecting hierarchical latent variable models. *Data Mining and Knowledge Discovery*, 33(4):1017–1058, 2019. 24

[81] Hoff PD. Dyadic data analysis with amen. *arXiv preprint arXiv:1506.08237*, 2015. 24

[82] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Fairness through causal awareness: Learning causal latent-variable models for biased data. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 349–358. ACM, 2019. 24

[83] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual review of statistics and its application*, 6:355–378, 2019. 24

[84] Tsitsulin A, Mottin D, Karras P, and Müller E. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 539–548. International World Wide Web Conferences Steering Committee, 2018. 25, 36, 39, 77, 128

[85] Ou M, Cui P, Pei J, Zhang Z, and Zhu W. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international*

conference on Knowledge discovery and data mining, pages 1105–1114. ACM, 2016. 25, 30, 36, 77, 128

[86] Rozemberczki B, Davies R, Sarkar R, and Sutton C. Gemsec: Graph embedding with self clustering. *arXiv preprint arXiv:1802.03997*, 2018. 25, 77, 128

[87] Rozemberczki B and Sarkar R. Fast sequence based embedding with diffusion graphs. In *International Conference on Complex Networks*, 2018. 25, 77, 128

[88] Yang Z, Cohen WW, and Salakhutdinov R. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016. 25, 77, 128

[89] Perozzi B, Al-Rfou R, and Skiena S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014. xiv, 25, 26, 34, 36, 77, 83, 128

[90] Grover A and Leskovec J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016. 25, 34, 36, 77, 128

[91] Sheikh N, Kefato Z, and Montresor A. gat2vec: representation learning for attributed graphs. *Computing*, 2018. 25, 77, 128

[92] Mikolov T, Sutskever I, Chen K, Corrado GS, and Dean J. Distributed representations of words and phrases and their compositionality. In *Adv Neur In*, pages 3111–3119, 2013. 25, 34, 36, 77, 128

[93] Cao S, Lu W, and Xu Q. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015. 25, 30, 36, 77, 128

[94] Liu Q, Li Z, Lui J, and Cheng J. Powerwalk: Scalable personalized pagerank via random walks with vertex-centric decomposition. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 195–204. ACM, 2016. 25, 77, 128

[95] Pandhre S, Mittal H, Gupta M, and Balasubramanian Vineeth N. Stwalk: learning trajectory representations in temporal graphs. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 210–219. ACM, 2018. 25, 77, 128

[96] Mikolov T, Chen K, Corrado G, and Dean J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 25, 77, 128

[97] Lin Y, Liu Z, Sun M, Liu Y, and Zhu X. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187, 2015. 25, 77, 128

[98] Wang Z, Ye X, Wang C, Wu Y, Wang C, and Liang K. Rsdne: Exploring relaxed similarity and dissimilarity from completely-imbalanced labels for network embedding. *Network*, 11:14, 2018. 25, 77, 128

[99] Justin Wang, Raymond KW Wong, and Thomas CM Lee. Locally linear embedding with additive noise. *Pattern Recognition Letters*, 2019. 29

[100] Zhang Jingjing. Identification of moving loads using a local linear embedding algorithm. *Journal of Vibration and Control*, page 1077546319833137, 2019. 29

[101] Liao L, He X, Zhang H, and Chua TS. Attributed social network embedding. *arXiv preprint arXiv:1705.04969*, 2017. 31

[102] Liang J, Jacobs P, Sun J, and Parthasarathy S. Semi-supervised embedding in attributed networks with outliers. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 153–161. SIAM, 2018. xiv, 33

[103] Yang C, Liu Z, Zhao D, Sun M, and Chang EY. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117, 2015. 34, 36

[104] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. 35, 36

[105] Chen CM, Tsai MF, Lin YC, and Yang YH. Query-based music recommendations via preference embedding. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 79–82. ACM, 2016. 36

[106] Dai H, Dai B, and Song L. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*, pages 2702–2711, 2016. 36

[107] Narayanan A, Chandramohan M, Chen L, Liu Y, and Saminathan S. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv preprint arXiv:1606.08928*, 2016. 36

[108] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 36

[109] Jihun Oh, Kyunghyun Cho, and Joan Bruna. Advancing graphsage with a data-driven node sampling. *arXiv preprint arXiv:1904.12935*, 2019. 38

[110] Zhihao Jia, Sina Lin, Rex Ying, Jiaxuan You, Jure Leskovec, and Alex Aiken. Redundancy-free computation graphs for graph neural networks. *arXiv preprint arXiv:1906.03707*, 2019. 38

[111] Gerome Vivar, Hendrik Burwinkel, Anees Kazi, Andreas Zwergal, Nassir Navab, and Seyed-Ahmad Ahmadi. Multi-modal graph fusion for inductive disease classification in incomplete datasets. *arXiv preprint arXiv:1905.03053*, 2019. 38

[112] Tang J, Qu M, and Mei Q. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174. ACM, 2015. 38

[113] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019. xiv, xvii, 39, 40, 41, 42

[114] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018. xiv, xvii, 39, 40, 41, 42

[115] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 41

[116] Yifan Hou, Hongzhi Chen, Changji Li, James Cheng, Ming-Chang Yang, and Fan Yu. A representation learning framework for property graphs. 2019. 41

[117] Guoli Yang, Cheng Zhu, and Weiming Zhang. Adaptive and probabilistic strategy evolution in dynamical networks. *Physica A: Statistical Mechanics and its Applications*, 518:99–110, 2019. 43

[118] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014. 48, 49, 50

[119] Mira A Amati V, Lomi A. Social network modelling. *Annu Rev Stat Appl*, 5, 2018. 76

[120] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000. 77

[121] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016. 94

[122] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. xvi, 97, 115

[123] François Chollet et al. Keras. `https://keras.io`, 2015. 99, 119

[124] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011. 115

[125] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 115