Representation Learning for Graph Structured Data using Deep Neural Networks

by

Uday Shankar Shanthamallu

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

Approved August 2021 by the
Graduate Supervisory Committee:

Andreas Spanias, Chair
Jayaraman J. Thiagarajan
Cihan Tepedelenlioglu
Visar Berisha

ARIZONA STATE UNIVERSITY

December 2021

ABSTRACT

Dealing with relational data structures is central to a wide-range of applications including social networks, epidemic modeling, molecular chemistry, medicine, energy distribution, and transportation. Machine learning models that can exploit the inherent structural/relational bias in the graph structured data have gained prominence in recent times. A recurring idea that appears in all approaches is to encode the nodes in the graph (or the entire graph) as low-dimensional vectors also known as embeddings, prior to carrying out downstream task-specific learning. It is crucial to eliminate hand-crafted features and instead directly incorporate the structural inductive bias into the deep learning architectures.

In this dissertation, deep learning models that directly operate on graph structured data are proposed for effective representation learning. A literature review on existing graph representation learning is provided in the beginning of the dissertation. The primary focus of dissertation is on building novel graph neural network architectures that are robust against adversarial attacks. The proposed graph neural network models are extended to multiplex graphs (heterogeneous graphs). Finally, a relational neural network model is proposed to operate on a human structural connectome. For every research contribution of this dissertation, several empirical studies are conducted on benchmark datasets. The proposed graph neural network models, approaches, and architectures demonstrate significant performance improvements in comparison to the existing state-of-the-art graph embedding strategies.

DEDICATION

To my family

ACKNOWLEDGEMENTS

Over the past several years, I have grown both professionally and personally, and I need to thank a number of people for my growth. I owe an outstanding debt of gratitude to my advisors Dr. Andreas Spanias and Dr. Jayaraman Thiagarajan, for their constant support and guidance. This Ph.D. would not have been possible without their encouragement and feedback. I am eternally thankful to them for inscribing good research skills in me. I am grateful to Dr. Cihan Tepedelenlioglu and Dr. Visar Berisha for their valuable time serving on my defense committee and their insightful comments and helpful feedback.

Over the course of this Ph.D. I was fortunate to interact with Michael Stanley, who has been an exceptional mentor. During my internship at NXP semiconductors, Michael taught me that even bad results are significant results. I am grateful to Lawrence Livermore National Laboratory and its scientists for providing me an opportunity to collaborate and work on advancing the research in graph representation learning. The work I have done in collaboration with them forms the essence of this dissertation.

I would like to recognize the invaluable assistance provided to me by the graduate advisors and the School of Electrical, Computer and Energy Engineering at ASU. I would like to thank my friends, peers, and colleagues at the SenSIP center for their kind support, frequent discussions, and memorable days in the lab. I also like to thank SenSIP industry partners for helping me develop my presentation skills.

Most importantly, none of this could have happened without my family. Their unconditional love and support have helped me achieve a great deal of success. In particular, I like to thank my mother Renuka and my wife Chinmayi Lanka for motivating me and sacrificing countless hours of their own time to help me accomplish my goals.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Graphs form a ubiquitous data structure and offer a distinct advantage in modeling many real-world relational data. Graphs can inherently capture the rich relational information between several entities and incorporate complex pairwise relationships between them. In the simplest form, a graph data structure is a collection of entities or objects represented as nodes (or vertices), along with pairwise relationships represented as edges. For example, individuals in a social network are represented as nodes in graphs, and their friendship is encoded by the set of edges as shown in Figure 1.1.

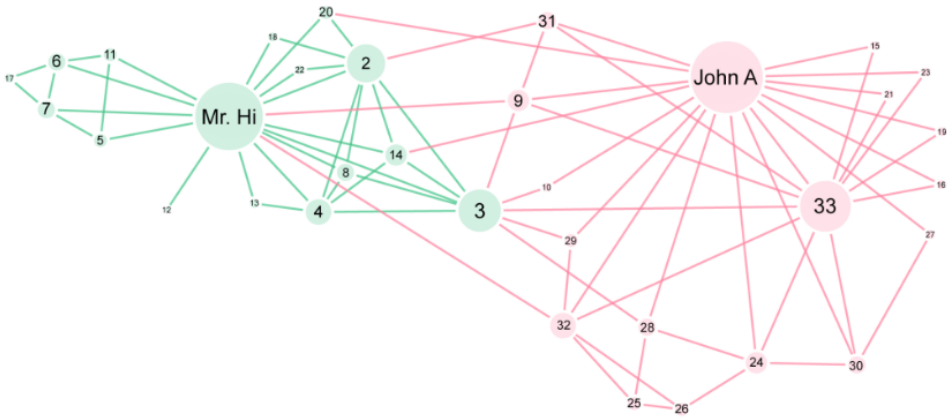


Figure 1.1: Zackary's Karate Club Social Network: 34 Members of the Social Network and Their Interactions Visualized as a Graph. The Social Network Was Split into Two Groups Due to a Conflict. The Coloring Scheme Identifies the Clustering of the Two Groups. Image Obtained From Jill Marie Hackett (Hackett, 2019).

Several other real-world data expressed as graphs are listed as follows:

- Social Networks (Eagle and Pentland, 2006) - Here, vertices/nodes are people, and edges represent interactions between them.
- Technological Network - The Internet is the worldwide network of physical connections between computers and other devices.
- Transportation Networks (Henderson *et al.*, 2012) - Examples include airline networks, road and rail networks
- Epidemic Networks(Simon *et al.*, 2011) - Modeling spreading of infectious diseases.
- Biological Networks - Human Brain connectome (Sporns *et al.*, 2005), Interactions at the cellular level, genome level, and protein level.
- Molecular Networks - Atoms represent nodes, and edges represent chemical bonds (Irwin *et al.*, 2012).

In scenarios where data is not readily available in a graph structure, a graph is constructed from the available data. The data is often assumed to be independent and identically distributed (*i.i.d*). Graph construction is performed without any supervision. *i.e.* label information is not required while constructing the graphs. An appropriate similarity metric that incorporates the domain knowledge is used to construct a graph from data instances. Two common construction methods are the *k-nearest neighbor* method and the ϵ *neighborhood* method. In the k-nearest neighbor method, for a given node (a data sample), connections are established only to the k nearest neighbors, where the closeness is quantitatively defined using a similarity metric (for example, Gaussian kernel) or distance function (for example, Euclidean distance). In ϵ- neighborhood-based graph construction, an undirected edge between

two nodes is added if the distance between them is smaller than ϵ, where $\epsilon > 0$ is a predefined constant.

1.1 Notation Summary

In this section, the notation used throughout this dissertation is summarized. The notation followed here is similar to those seen in deep learning books (Goodfellow *et al.*, 2016).

x	A scalar
$\mathbf{x}$	A vector
$\mathbf{X}$	A matrix
x_i,	The i^{th} element of vector $\mathbf{x}$
X_{ij},	The element of matrix $\mathbf{X}$ at row i and column j
$\mathcal{X}$	A set
$\lvert\mathcal{X}\rvert$	Cardinality of the set $\mathcal{X}$
$\mathbb{R}^n$	The set of n-dimensional vectors of real numbers
$f(\cdot)$	A function
$\langle \mathbf{x}, \mathbf{y} \rangle$	Dot product or inner product of vectors $\mathbf{x}$ and $\mathbf{y}$
$P(\cdot)$	Probability distribution
$p(x)$	Probability density function
$z \sim P(\cdot)$	Random variable z has probability distribution P or z is sampled from P
M	A neural network model
$\boldsymbol{\Theta}$	Parameters of the neural network model
σ	Non-linearity function (Sigmoid, tanh, ReLU etc)
$\mathcal{O}$	Big O notation

1.2 Types of Graphs

Formally, a simple graph is represented by the tuple $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes with cardinality $|\mathcal{V}| = N$, and $\mathcal{E}$ denotes the set of edges with cardinality $|\mathcal{E}| = M$. Optionally, the complete graph can be represented by a $N \times N$ symmetric adjacency matrix $\mathbf{A}$ where $A_{ij} \in \{1, 0\}$, $\mathbf{A} \in \mathbb{R}^{N \times N}$. The value of A_{ij} is 1 if there is an edge connecting the nodes v_i and v_j , otherwise 0 (thus $A_{ij} = A_{ji}$). Additionally, the edges can also encode strength and symmetry in the relationships.

Directed Graph: If the edges in the graphs are directed, that is the relationship is not mutual then the adjacency matrix $\mathbf{A}$ is no longer symmetric ($A_{ij} \neq A_{ji}$). For example, the Twitter network is based on follower-ship, and hence it is a directed network.

Weighted Graph: For weighted graphs, the edges also encode the strength of connectivity of the relationship. A weighted adjacency matrix $\mathbf{W}$ is used in place of $\mathbf{A}$. As before, if there is an edge $e = (i, j)$ connecting nodes i and j, the entry W_{ij} represents the weight associated with the edge; otherwise $W_{ij} = 0$. For example, a human brain connectome is represented by a weighted graph, where the edges represent the strength of the structural connectivity between different voxels(regions) of the brain.

Multi-layer graph: A multi-layered graph is represented using a set of L inter-dependent graphs $\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)}), \text{for } l = 1, \ldots, L$, where there exists a node mapping between every pair of layers to indicate which vertices in one graph correspond to vertices in the other. In our setup, we assume $\mathcal{V}^{(l)}$ from all layers contain the same set of nodes, while the edge sets $\mathcal{E}^{(l)}$ (each of cardinality $M^{(l)}$) are assumed to be different.

Degree of a node: The number of neighbors of a node v is called the degree of v and is denoted by $d(v)$; $d(v_i) = \sum_j A_{ij}$ for unweighted graphs and $d(v_i) = \sum_j W_{ij}$

for weighted graphs.

Graph Signal or Node Attribute: In addition to the network structure, each node v_i can be endowed with a set of attributes, $\mathbf{x}_i \in \mathbb{R}^D$, $i \in [N]$. For efficient graph representation learning, one must incorporate the node attributes in addition to structural information.

1.3 Machine Learning on Graphs

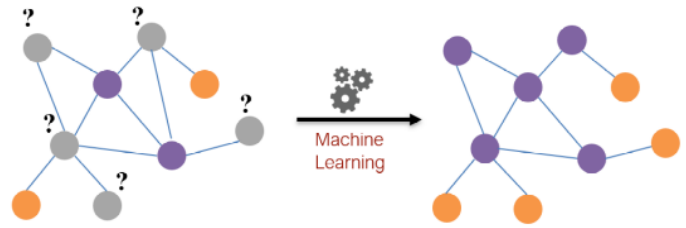
Although supervised and unsupervised machine learning (Shanthamallu *et al.*, 2017) are popular for other data structures such as images, machine learning problems on graphs often blur the boundaries between the traditional machine learning categories. Some of the typical machine learning tasks are discussed here.

1.3.1 Node Classification

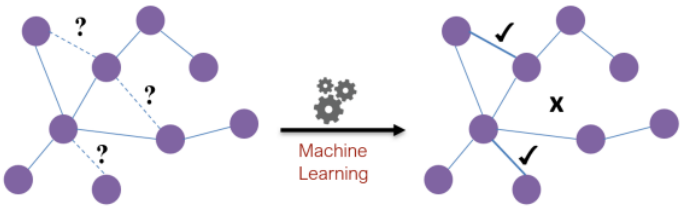
In many real-world graphs, nodes are associated with helpful information, often treated as labels of these nodes. For example, in social networks, such information can be demographic properties of users such as age, gender, occupation, or users' interests and hobbies. The goal of node classification is as described follows: given the ground truth for a small set of nodes, predict the labels for the rest of the nodes in the graph.

Definition 1.3.1 *Node classification: Given a graph $G = (\mathcal{V}, \mathcal{E})$ and labels for the subset of nodes $\mathcal{V}_{label} \subset \mathcal{V}$, predict the labels for each of the nodes in the set $v \in \mathcal{V} \setminus \mathcal{V}_{label}$.*

The goal of the node classification task is to learn a mapping function by leveraging the graph structure G and labels of nodes in $\mathcal{V}_{label}$. Note that $\mathcal{V}_{label} \cup \mathcal{V}_{unlabel} = \mathcal{V}$ and $\mathcal{V}_{label} \cap \mathcal{V}_{unlabel} = \emptyset$ where $\mathcal{V}_{unlabel} = \{v \in \mathcal{V} \setminus \mathcal{V}_{label}\}$. Figure 1.2a shows an example



(a) Node Classification



(b) Link Prediction

Figure 1.2: Typical Machine Learning Tasks on Graph Data Structures: (A) Node Classification: The Goal Is to Predict the Labels of the Unlabeled Nodes given a Few Labeled Nodes. (B) Link Completion: The Goal Is to Predict the Missing Links given the Incomplete Graph Connections.

of node classification (or node-label prediction) where we know the labels for a small subset of nodes.

1.3.2 Link Prediction

While node classification is helpful in inferring information of a node based on its relationship with other nodes in the graph, the goal of the link prediction is to infer the missing relationships (edges) in the graph. For example, if we know some protein-protein interaction happening inside a biological cell, is it possible to infer the unseen or unknown interactions? Link prediction is often referred to by other terms such as relation prediction, graph completion, or recommendation, depending on the application.

Definition 1.3.2 *Link Prediction: Given an incomplete graph G with a set of nodes $\mathcal{V}$ and an incomplete set of edges between these nodes $\mathcal{E}_{train} \subset \mathcal{E}$, infer the missing edges $\mathcal{E} \setminus \mathcal{E}_{train}$.*

Figure 1.2b shows a typical example of link prediction. By utilizing the available partial information, new interactions can be identified.

1.3.3 Graph Classification and Regression

Similar to the node classification task, the graph classification task deals with identifying the labels for the entire graph rather than the individual nodes. For example, consider a chemical molecule represented as a $2D$ graph; one can build a machine learning model to predict the biological activity (actively binding or not) to a specific receptor. Similar to graph level classification, one can also build a regression model over the entire graph (for example, identifying the solubility of a molecule).

Definition 1.3.3 *Graph classification: Given a set of T training graphs $\{(G_t, y_t)\}_{t=1}^{T}$ where y_i is the label of the graph G_i and y_i assumes one of the C pre-defined classes, the goal is to learn a mapping function $f : \mathcal{G} \mapsto \mathcal{Y}$ where $\mathcal{G}$ is in the input graph space and $\mathcal{Y}$ is the output space.*

In graph classification or regression tasks, the central idea is to learn over the entire graph data, but instead of making node-level predictions, we are instead given a dataset of multiple different graphs, and our goal is to make independent predictions specific to each graph. That is, we want to learn a mapping function over graphs instead of nodes.

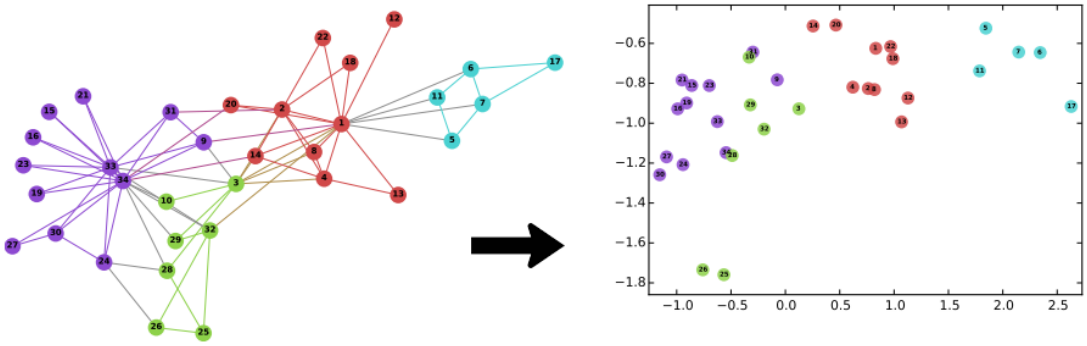


Figure 1.3: Learning Node Embeddings: Random Walk Based Node Embedding Technique Applied on Zachary's Karate Club Graph. Nodes in the Graph Have Distinct Labels [Left] and the Learned 2-dimensional Node Embeddings Preserve the Community Structure [Right]. Image Obtained from (Perozzi *et al.*, 2014).

1.4 Node Embeddings

Despite the variability in graph-based machine learning tasks, a recurring idea that appears in almost all of these applications is to obtain low-dimensional vectors, also known as embeddings for nodes, edges, or even the entire graph, before carrying out the downstream learning task. We want those embeddings to preserve the structural information or the local neighborhood information (relationships). This is the fundamental premise in graph-based machine learning and graph representation learning, *i.e.,* to learn to represent or encode the graph structure into embeddings so that the downstream task-specific machine learning models can exploit it. An example of such embedding algorithm on Zachary's karate club graph (Girvan and Newman, 2002) is shown in Figure 1.3.

In the simplest form, the adjacency matrix indicating the connectivities can be treated as na "ive embeddings for the nodes. However, it is well known that such cursed, high-dimensional representations can be ineffective for subsequent learning.

Hence, there has been a long-standing interest in constructing low-dimensional embeddings that best represent the network topology. However, the main challenge in graph-based machine learning is that the structure of the graph/network is irregular when compared with images, or audio, or text. Images, speech, and other time-series data are structured by Euclidean grids. On the other hand, graphs are non-Euclidean; simple operations like convolution, translation, and downsampling can not be explicitly defined on graphs.

Traditional graph-based machine learning approaches make use of graph Laplacian structure and regularization methods that involve matrix decomposition - examples include Spectral Clustering (Ng *et al.*, 2002), stochastic factorization of the adjacency matrix (Ahmed *et al.*, 2013), decomposition of the modularity matrix (Newman, 2006; Chen *et al.*, 2014) etc. The unprecedented success of deep learning with data defined on regular domains, e.g., images and speech, has motivated its extension to arbitrarily structured graphs. For example, Yang *et al.* (Yang *et al.*, 2016) and Thiagarajan *et al.* (Thiagarajan *et al.*, 2016) have proposed stacked auto-encoder style solutions that directly transform the objective measure into an undercomplete representation. An alternate class of approaches utilize the distributional hypothesis, popularly adopted in language modeling (Harris, 1954), where co-occurrence of two nodes in short random walks implies a strong notion of semantic similarity to construct embeddings – examples include *DeepWalk* (Perozzi *et al.*, 2014) and *Node2Vec* (Grover and Leskovec, 2016).

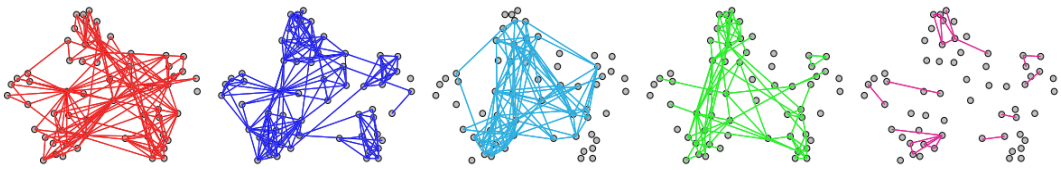
While the approaches above effectively preserve network structure, semi-supervised learning with graph-structured data requires feature learning from node attributes to effectively propagate labels to unlabeled nodes. Since convolutional neural networks (CNNs) have been the mainstay for feature learning with data defined on regular grids, the natural idea is to generalize convolutions to graphs. Existing work on

this generalization can be categorized into *spectral* approaches (Bruna *et al.*, 2013; Defferrard *et al.*, 2016a), which operate on an explicit spectral representation of the graphs, and *non-spectral* approaches that define convolutions directly on the graphs using spatial neighborhoods (Duvenaud *et al.*, 2015; Niepert *et al.*, 2016). More recently, *attention* models (Velickovic *et al.*, 2017) have been introduced as an effective alternative for graph data modeling which mainly rely on attention mechanisms for feature learning.

Graph Attention Networks (GATs): The attention mechanism in neural networks focuses on the most relevant part of the data to make decisions. In contrast to graph spectral approaches, graph attention models do not require the construction of an explicit Laplacian operator and can be readily applied to non-Euclidean data. Velickovic *et al.,* make use of attention heads to perform node classification in semi-supervised learning. An attention head parameterizes the local dependencies to determine the most relevant parts of the graph to focus on while computing the features for a node. Further, GATs are easily scalable to large networks. Details of attention head and its construction is provided in Chapter 2.

1.4.1 Node Embeddings for Multi-layer Graphs

Modern data analysis pipelines are becoming increasingly complex due to the presence of multi-view information sources. While graphs are effective in modeling complex relationships, in many scenarios, a single graph is rarely sufficient to represent all interactions succinctly, and hence multilayered graphs have become popular. For example, in a social network setting, multiple aspects of relationships can be represented by a multilayer graph comprised of multiple interdependent graphs. Each graph represents an aspect of the relationships. One such network is shown in Figure 1.4 which is obtained from the AUCS dataset (Kim and Lee, 2015). Here, multi-



(a) Work. (b) Lunch. (c) Facebook. (d) Friend. (e) Coauthor.

Figure 1.4: AUCS Dataset: An Example of Multi-Layer Graphs. Image Obtained From (Kim *et al.*, 2016).

ple layers represent different aspects of the relationship among 61 employees at a University.

Though multilayer graphs lead to richer representations, extending solutions from the single-graph case is not straightforward. Until recently, the majority of existing work has focused on analysis and inferencing from a single network. Consequently, there is a strong need for novel solutions to solve classical problems, such as node classification, in the multilayered case. Our definition of multilayered graphs assumes complementary views of connectivity patterns for the same set of nodes, thus requiring the need to model complex dependency structures across the views. The heterogeneity in the relationships, while providing richer information, makes statistical inferencing challenging. Note that alternative definitions for multi-view networks exist in the literature (Li *et al.*, 2018), wherein the node sets can be different across layers (e.g., interdependent networks).

1.5 Problem Statement

1.5.1 Study and Analysis of Attentions Inferred by GATs

With the widespread adoption of attention models in language modeling and computer vision, it has become imperative to study and understand the functioning and

robustness of attention mechanisms. Though GATs have successfully achieved state-of-the-art performance in semi-supervised node classification, a detailed analysis of the attention mechanism is not yet available. In particular, the robustness of the attention mechanism in the presence of adversaries(for example, noisy nodes) needs to be studied.

1.5.2 Building Robust GNN Models to Defend Adversarial Attacks

Graph Neural Networks (GNNs), a generalization of neural networks to graph-structured data, are often implemented using message passes between entities of a graph. While GNNs are effective for node classification, link prediction, and graph classification, they are vulnerable to adversarial attacks. GNNs inherit both advantages and disadvantages of DNNs. A small perturbation to the structure can lead to non-trivial performance degradation. The adversary can generate graph adversarial perturbations by manipulating the graph structure or node features to fool the GNN models. This limitation of GNNs has arisen immense concerns on adopting them in safety-critical applications such as financial systems and risk management.

1.5.3 Efficient Representation Learning for Multi-Layer Graphs

Multilayer graphs can encode multiple aspects of relationships compared to a simple graph. In spite of having rich information, it is challenging to learn vector representations by effectively combining information from both with-in-layer connections and cross-layer connections. A vast majority of existing work in graph-based semi-supervised learning is concentrated on single-layered graphs. Is it possible to extend existing tools to a multiplex-graph?

1.5.4 Learning Node Embeddings from Random Features

One of the key assumptions in the recently proposed graph-spectral approaches (Defferrard *et al.*, 2016b) is the availability of the node attributes in addition to the network structure. Latent representations for nodes are learned in a task-specific manner, where node attributes are used along with the graph structure to propagate labels to unlabeled nodes effectively. In practice, graph datasets often comprise only the edge sets without any additional information (especially in multilayer datasets). There is a need to employ a randomized initialization strategy for creating node attributes.

1.5.5 Application of Graph Neural Networks to Healthcare Data

For biological data such as the data from the human brain connectome, many classical approaches have relied on hand-engineering statistical descriptors from structural or functional connectomes to build predictive models. However, there is growing interest in leveraging deep learning techniques. Though the human connectome is often viewed as a graph defined with each node indicating to a brain region, and the edges representing neural connections, we argue that existing graph neural network solutions that are built on the assumption of information diffusion are not directly applicable.

1.6 Contributions

- Qualitative and Quantitative analysis of attention mechanisms in the Graph attention networks is provided in detail. The vulnerability of the existing attention models in the presence of adversaries (structural noise) is discussed in (Shanthamallu *et al.*, 2020). A robust GAT variant is proposed that analyzes

the distribution of the attention coefficients. Performance improvements of the Robust GAT are demonstrated with experiments using citation networks.

- Uncertainty Matching GNN (`UM-GNN`) architecture (Shanthamallu *et al.*, 2021) is proposed for improving the robustness of GNN models, particularly against poisoning attacks to the graph structure (Chapter 3). `UM-GNN` leverages epistemic uncertainties from the message passing framework. More specifically, we propose to build a surrogate predictor that does not directly access the graph structure but systematically extracts reliable knowledge from a standard GNN through a novel uncertainty-matching strategy. Interestingly, this uncoupling makes `UM-GNN` immune to evasion attacks by design and achieves significantly improved robustness against poisoning attacks. Using empirical studies with standard benchmarks and a suite of global and target attacks, we demonstrate the effectiveness of `UM-GNN` when compared to existing baselines, including the state-of-the-art robust GCN.

- Attention models are developed for multilayered graphs in semi-supervised learning problems (Shanthamallu *et al.*, 2019b). Two architectures are proposed that perform layer-wise attention modeling and fuses information from different layers. Proposed approaches are evaluated on several benchmark datasets and show that they outperform existing network embedding strategies (Chapter 4).

- Randomized initialization strategy is employed for creating node attributes (Shanthamallu *et al.*, 2019b). Random initialization has been highly successful in making word representations for Natural Language Programming tasks, and in many scenarios, its performance matches or even surpasses pre-trained word embeddings (Chapter 4).

- We develop a structured network architecture termed (RGNN) (Shanthamallu *et al.*, 2019a) that uses the connectome to constrain the message passing between two network layers representing edges and nodes, respectively (Chapter 5). Using connectomes from the Human Connectome Project (HCP), we show that the proposed approach can effectively predict meta-information such as age and gender, and accurately recover the volumes of different brain regions, which are known to be encoded in the connectomes.

1.7 List of Publications

In this section, I provide the list of publications that have allowed me to publish my research contributions in graph representation learning space, including adversarial attacks and defense on GNN models:

- Uday Shankar Shanthamallu, Jayaraman J. Thiagarajan, and Andreas Spanias. "A regularized attention mechanism for graph attention networks." ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). IEEE, 2020.

- Uday Shankar Shanthamallu, Jayaraman J. Thiagarajan, and Andreas Spanias. "Uncertainty-Matching Graph Neural Networks to Defend Against Poisoning Attacks." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35. No. 11, February 2021.

- Uday Shankar Shanthamallu, Jayaraman J. Thiagarajan, Huan Song, Andreas Spanias. "Gramme: Semisupervised learning using multilayered graph attention models". IEEE Transactions on neural networks and learning systems, 31(10), 3977-3988, November 2019. U.S. Non-Provisional Patent Application Serial No. 16/739,824.

- Uday Shankar Shanthamallu, Qunwei Li, Jayaraman J. Thiagarajan, Rushil Anirudh, Timo Bremer. "Modeling Human Brain Connectomes using Structured Neural Networks". NeurIPS workshop on Graph Representation Learning, December 2019.

- Uday Shankar Shanthamallu, Andreas Spanias, Cihan Tepedelenlioglu, Mike Stanley. "A brief survey of machine learning methods and their sensor and IoT applications." 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA). IEEE, August 2017.

1.8 Organization of the Dissertation

The organization of the rest of the dissertation is given below. Chapter 2 provides a detailed literature review on graph representation learning methods in the context of graph-based machine learning. It covers a multitude of embedding techniques including Laplacian (Belkin and Niyogi, 2003), shallow embedding techniques like Node2Vec (Grover and Leskovec, 2016), and graph neural networks (Wu *et al.*, 2019b). Chapter 3 highlights the vulnerabilities of GNN models to adversarial attacks and a novel architecture is introduced to defend against such attacks. Chapter 4 deals with approaches for extending graph neural networks to multivew graphs which consist of more than one aspect of relationships. These multi-layer graphs are composed of many adjacency matrices for representing the relationships. Chapter 5 deals with the application of graph neural networks to biological data such as human connectome. Chapter 6 provides the concluding remarks of this dissertation and summarizes future research directions.

Chapter 2

LITERATURE REVIEW ON GRAPH REPRESENTATION LEARNING

In this chapter, prior work on graph representation learning in the context of machine learning is reviewed. Representation learning approaches on graphs can be broadly classified into three distinct methods: matrix factorization, encoder-decoder including random walk, and graph neural networks. We begin with the traditional representation learning approaches for single-layer graphs that use graph Laplacian embedding, regularization methods, and matrix factorization. We then discuss several shallow embedding approaches that learn low-dimensional vector embeddings for graph structure in an unsupervised way. Next, we introduce graph neural network approaches that use graph structure either directly or indirectly in neural networks. We then discuss the attention model on graphs that use attention head units to learn hidden representations for each node. Lastly, we review the existing methodologies that can be applied to multi-layer graph analysis.

2.1 Matrix Factorization Methods

Since many of the real-world networks are rather sparse, embedding techniques involved matrix factorization of graph adjacency matrix to find an appropriate low-rank matrix for the original graph. The factorization method corresponds to the structure-preserving dimensionality reduction process. In spectral graph theory, the spectral properties of the graphs are studied via matrix decomposition of the associated graph matrices such as *adjacency matrix*- $\mathbf{A}$ and the *graph Laplacian matrix*- $\mathbf{L}$ and its variants. In particular, the graph properties are understood through the lens of eigenvalues and eigenvectors obtained from the associated graph matrices.

The Laplacian allows a natural link between discrete representations (graphs) and continuous representations (vector spaces). As an example, spectral clustering is the solution obtained for the graph partitioning problem. The graph Laplacian Matrix is obtained from adjacency matrix and the degree matrix $\mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix with the degree of each node as its diagonal elements. The Laplacian matrix is obtained as

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{2.1}$$

$\mathbf{L}$ is generally referred to as unnormalized graph Laplacian. There are two other normalized versions of the Laplacian:

- Symmetric normalized Laplacian

$$\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$$

- Random walk Laplacian

$$\mathbf{L}_{RW} = \mathbf{D}^{-1}\mathbf{L}$$

Given any variant of graph Laplacian, the k dimensional embedding for each node is obtained by first performing Eigen-decomposition of the Laplacian and selecting the first k Eigenvectors (corresponding to the k smallest Eigenvalues). Laplacian eigenmaps (Belkin and Niyogi, 2003) is one of the popular methods that build upon the spectral clustering approach.

2.2 Encoder-Decoder Methods

In this section, node embedding techniques that are based upon the encoder-decoder framework are discussed. In this style of learning, the encoder maps the nodes of the graph into a low-dimensional vector space (*a.k.a* embedding) as seen in Figure 2.1. Then a decoder model takes the low-dimensional node embeddings and

uses them to reconstruct information about each node's neighborhood in the original graph. Note that the simplest form of such an encoder can be a look-up table.

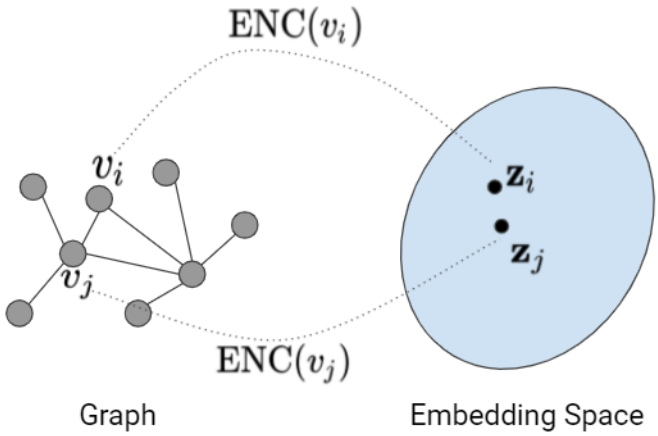


Figure 2.1: Encoder-Decoder Approach for Learning Node Embeddings. The Encoder (Enc) Maps the Nodes into a Low Dimensional Space Such That the Nearby Nodes in the Graph Have Short Distances in the Embedding Space.

More formally,

$$\text{ENC} : \mathcal{V} \rightarrow \mathbb{R}^d$$

The embeddings for all the nodes in the graph can be represented by the matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$. On the other hand, the decoder tries to reconstruct the graph from the embeddings generated by the encoder. For example, one can employ a pairwise decoder to reconstruct the edge (if any) connecting v_i and v_j from the embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$ respectively. Formally,

$$\text{DEC} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$$

$$\text{DEC}\big(\text{ENC}(v_i), \text{ENC}(v_j)\big) = \text{DEC}(\mathbf{z}_i, \mathbf{z}_j) \approx \mathbf{A}_{ij} \tag{2.2}$$

The encoder-decoder framework can be optimized by minimizing the discrepancy between the decoder reconstruction objective given in equation 2.2 and the graph adjacency matrix. In summary, the encoder and decoder are trained so that the pairwise node relationships can be effectively encoded in the embeddings. Such encoder-decoder methods can be seen in GraRep(Cao *et al.*, 2015), HOPE (Ou *et al.*, 2016).

2.2.1 Random Walk Methods

Random-walk methods perform a large number of random walks of fixed length where the starting node is chosen randomly. These short random walks provide the strength of connectedness between different nodes. Nodes that frequently co-occur in a random walk should be closely embedded in the embedding space. Also, random walks can capture the global structure of the graphs in addition to the local structure. *DeepWalk* (Perozzi *et al.*, 2014) and *node2vec* (Grover and Leskovec, 2016) algorithms, which are random-walk based methods, utilizes neural optimization techniques originally designed for language modeling to obtain dense, low-dimensional embeddings for nodes in a graph. In language modeling, the goal is to learn a low-dimensional vector representation for natural words, based on the context, from a large corpus. The latent representations reveal rich semantic information and estimate the likelihood of a specific sequence of words appearing in the corpus. Similar to word embeddings in the NLP literature, the distances in latent dimensions provide a convenient metric for understanding similarities between nodes in the network. Furthermore, such continuous vector spaces enable the definition of smooth decision boundaries between different communities and groups with homogeneous behavior. Given a graph G and the binary adjacency matrix of size $|\mathcal{V}| \times |\mathcal{V}|$, the goal is to generate latent representations, $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where d is the number of dimensions of the embedding and $|\mathcal{V}|$ indicates the cardinality of the vertex set.

Formally, a random walk is a stochastic process with a set of random variables defined as vertices chosen at random from the neighbors of each vertex in the sequence. The ability of random walk to reveal the local structure makes it a natural tool for extracting information from graphs. Let us consider a random walk $\mathcal{W}_t$ in step t, which is rooted at the vertex v_i. The transition probability between the nodes v_i and v_j can be expressed as

$$P(\mathcal{W}_{t+1} = v_j | \mathcal{W}_t = v_i) = f(\|\mathbf{z}_i - \mathbf{z}_j\|_2) \tag{2.3}$$

where $\|\mathbf{z}_i - \mathbf{z}_j\|_2$ indicates the similarity metric between the two vertices in the latent space to be recovered, and f is a linking function that connects the vertex similarity to the actual co-occurrence probability. Interestingly, with an appropriate choice of length of the walks, the true metric can be recovered accurately from the co-occurrence statistics constructed using random walks. Furthermore, in (Perozzi *et al.*, 2014), the authors note that the frequency in which vertices appear in the short random walks follows a power-law distribution, similar to words in natural language corpora. This naturally motivates the use of ideas from neural language modeling. While DeepWalk performs an unbiased random walk based on the depth-first sampling, node2vec performs a biased random walk that allows visiting the previously visited nodes.

Though the encoder-decoder approaches have achieved many successes in representation learning for nodes, these shallow embedding techniques suffer some crucial drawbacks. Firstly, a unique embedding is learned for each node as the encoder does not share parameters. This makes the approach computationally inefficient as the number of parameters grows as $\mathcal{O}(|\mathcal{V}|)$. Secondly, the node attributes or the signal defined on each node are not incorporated while learning the embeddings. The rich feature information on the nodes can provide additional insights into the node

neighborhood structure.

2.3 Graph Neural Network Methods

Some of the limitations seen in the previous methods can be overcome by utilizing *graph neural networks* (GNN), an extension of neural networks for graph-structured data. The central idea of GNN formalism is to construct efficient representation for each node (or edge or graph) from the graph structure as well as node attributes (or edge attributes) by allowing parameter sharing. The fundamental premise of GNN lies in the *message passing* mechanism in which (i) node feature information is exchanged between nodes, (ii) messages are aggregated and transformed using a non-linearity using neural networks.

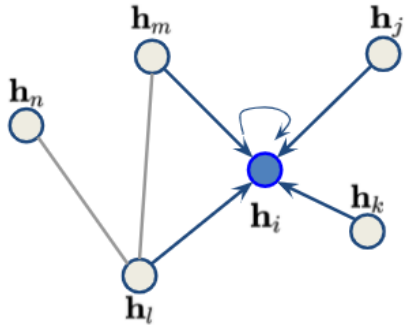


Figure 2.2: The Blue Node v_i Updates Its Feature by First Aggregating Messages from Its Neighboring Nodes (v_j, v_k, v_l, v_m) with the Help of a Message Function M and Then Transforms the Aggregated Messages Using an Update Function U.

2.3.1 Message Passing Framework

The message passing framework repeatedly applies the following two functions on every node of the graph.

- *Message Function*: The message function M is an arbitrary differentiable func-

tion (neural networks, for example). The message received by the node v_i is formally given as

$$\mathbf{m}_i = \sum_{j \in \mathcal{N}_i} M(\mathbf{h}_i, \mathbf{h}_j, e_{ij}) \tag{2.4}$$

Here, h_i, h_j denote the current hidden representations of node v_i and v_j respectively. In a nutshell, the message function generates a message based on the node's neighborhood.

- *Update Function*: The update function uses $\mathbf{h_i}$, the current representation of the node v_i, and the received message m_i to generate the transformed representation $\mathbf{h}_i^+$ as shown below:

$$\mathbf{h}_i^+ = U(\mathbf{h}_i, \mathbf{m}_i) \tag{2.5}$$

Note that the update function can also be any arbitrary differentiable function with learnable parameters.

An example of the message passing framework is given in Figure 2.2. The message passing framework can be applied repeatedly on each node, which facilitates messages from nodes other than the local neighborhood. In other words, if the message passing framework is applied 3 times, then the messages can be received from nodes in the 3-hop neighborhood.

2.3.2 Graph Convolutional Network

Given the success of convolutional neural networks(CNNs) in feature learning from data defined on regular grids (e.g., images), the next generation of graph neural networks focused on generalizing convolutional neural networks to graphs with node

attributes. Node attributes are nothing but the data defined on the graph at each vertex. Collectively, these are termed graph signals. An example of a graph with signals defined on the nodes is shown in Figure 2.3. The generalization of CNNs to graphs is not straightforward because convolution and pooling(downsampling) operations are defined for regular grids. In graph convolutional neural networks, the feature learning is carried out to transform signals defined at nodes into meaningful latent representations, akin to filtering of signals (Shuman *et al.*, 2013). The most challenging task in generalizing CNNs to graphs is defining a localized graph filter. CNNs extract the local stationarity property of the input data or signals by revealing local features that are shared across the data domain.

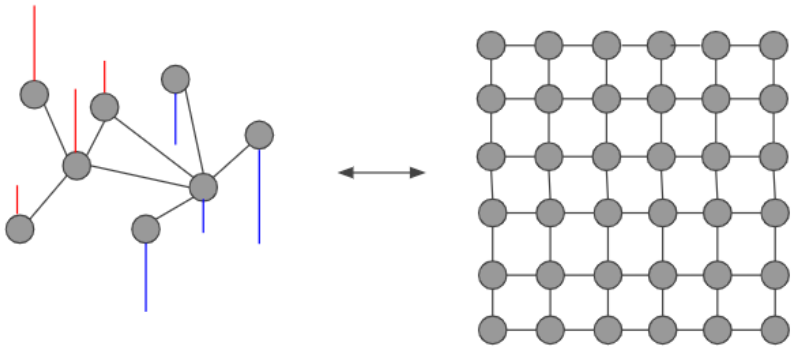


Figure 2.3: The Signal Defined on the Nodes of the Graph in Comparison to the Signals Defined on the Regular Euclidean Grid (Images). The Red Color Indicates That the Graph Signal Is Positive Where as Blue Color Indicates a Negative Value.

Since the spatial convolution operation cannot be directly defined on arbitrary graphs, various spectral domain and neighborhood-based techniques have been developed. As the name suggests, spectral approaches operate using the spectral representation of graph signals, defined using the eigenvectors of the graph Laplacian. For example, in (Bruna *et al.*, 2013), convolutions are realized as multiplications in the graph Fourier domain; however, since the filters cannot be spatially localized

on arbitrary graphs, this relies on the explicit computation of the spectrum based on matrix inversion. Consequently, special families of spatially localized filters have been considered. Examples include the localization technique in (Henaff *et al.*, 2015), and Chebyshev polynomial expansion based localization in (Defferrard *et al.*, 2016a). Building upon this idea, Kipf and Welling (Kipf and Welling, 2016) introduced graph convolutional neural networks (GCN) using a localized first-order approximation of spectral graph convolutions, wherein the filters operate within a one-step neighborhood, thus making it scalable to even large networks.

On the other hand, with non-spectral approaches, convolutions are defined directly on graphs and can work with different-sized neighborhoods. For example, localized spatial filters with different weight matrices for varying node degrees are learned in (Duvenaud *et al.*, 2015). Whereas, in approaches such as (Niepert *et al.*, 2016) neighborhood for each node is normalized to achieve a fixed size neighborhood. More recently, attention models, commonly used to model temporal dependencies in sequence modeling tasks, were generalized to model neighborhood structure in graphs. More specifically, graph attention networks (Velickovic *et al.*, 2017) employ dot-product-based self-attention mechanisms to perform feature learning in semi-supervised learning problems.

2.3.3 Graph Attention Models

In this section, we discuss in detail the recently proposed graph attention model (Velickovic *et al.*, 2017), which uses an attention mechanism to learn node embeddings. The attention mechanism is a widely adopted strategy in sequence-to-sequence modeling tasks. A parameterized function is used to determine relevant parts of the input to focus on to make decisions. A recent popular implementation of the attention mechanism in sequence models is the *Transformer* architecture by Vaswani *et al.*

(Vaswani *et al.*, 2017), which employs *scalar dot-product* attention to identify dependencies. Furthermore, this architecture uses a *self-attention* mechanism to capture dependencies within the same input and employs multiple *attention heads* to enhance the modeling power. These important components have been subsequently utilized in a variety of NLP tasks (Yang *et al.*, 2017; Barone *et al.*, 2017) and clinical modeling (Song *et al.*, 2017).

An attention head in the graph attention layer learns a latent representation for each node by aggregating the features from its neighbors. More specifically, the feature at a node is computed as the weighted combination of features from its neighbors, where the weights are obtained using the attention function. Following our notations, each node v_i is endowed with a $D-$dimensional attribute vector $\mathbf{x}_i$, and hence the input to graph attention layer is denoted by the set of attributes $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ or $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times D}$ denoted in matrix form. The attention layer subsequently produces $d-$dimensional latent representations $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_N\}$ represented in matrix form as $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$.

An attention head is constructed as follows: First, a linear transformation is applied to the features at each node, using a shared and trainable weight matrix $\mathbf{\Theta} \in \mathbb{R}^{d \times D}$, thus producing intermediate representations,

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{\Theta}^T \tag{2.6}$$

Subsequently, a scalar dot-product attention function is utilized to determine attention weights for every edge in the graph, based on features from the incident neighbors. Formally, the attention weight for the edge e_{ij} connecting the nodes v_i and v_j is computed by performing dot-product attention mechanism as shown below

$$a_{ij} = \langle \mathbf{a}, \tilde{\mathbf{x}}_i || \tilde{\mathbf{x}}_j \rangle \tag{2.7}$$

where $\mathbf{a} \in \mathbb{R}^{2d}$ denotes the parameters of the attention function, and $||$ represents

concatenation of the transformed features of nodes v_i and v_j respectively. The attention weights a_{ij} are computed with respect to every node in the neighborhood of v_i, i.e., for $v_j \in \mathcal{N}_i \cup \{v_i\}$, where $\mathcal{N}_i$ represents the neighborhood of v_i. Note that, we include the self-edge for every node while implementing the attention function. The weights are then normalized across all neighboring nodes using a softmax function, thus producing the normalized attention coefficients.

$$\alpha_{ij} = \texttt{Softmax}_j(a_{ij}) = \frac{\exp(a_{ij})}{\sum_{k \in \mathcal{N}_i \cup \{v_i\}} \exp(a_{ik})} \tag{2.8}$$

Finally, the normalized attention coefficients are used to compute the latent representation at each node, through a weighted combination of the node features. Note that a non-linearity function σ is also utilized at the end to improve the approximation.

$$\mathbf{z}_i = \sigma\left(\sum_{j \in \mathcal{N}_i \cup \{v_i\}} \alpha_{ij} \tilde{\mathbf{x}}_j \right) \tag{2.9}$$

An important observation is that the attention weights are not required to be symmetric. For example, if a node v_i has a strong influence on node v_j, it does not imply that node v_j also has a strong influence on v_i and hence $\alpha_{ij} \neq \alpha_{ji}$. The operations from equations (2.6) to (2.9) constitute a single head. While this simple parameterization enables effective modeling of relationships in a graph while learning latent features, the modeling capacity can be significantly improved by considering multiple attention heads. Following the *Transformer* architecture (Vaswani *et al.*, 2017), the output latent representations from the different heads can be aggregated using either concatenation or averaging operations.

2.4 Multi-layered Graph Analysis

Prior work on multi-layered graphs focuses extensively on unsupervised community detection and they can be broadly classified into methods that obtain a consensus

community structure for producing node embeddings (Dong *et al.*, 2012), (Dong *et al.*, 2014), (Kim *et al.*, 2017), (Tagarelli *et al.*, 2017), and methods that infer a separate embedding for a node in every layer while exploiting the inter-layer dependencies, and produce multiple potential community associations for each node (Mucha *et al.*, 2010), (Bazzi *et al.*, 2016).

Analysis and inferencing with multi-layered graphs is a challenging yet crucial problem in data mining. With each layer characterizing a specific kind of relationship, the multi-layered graph comprehensively represents every type of relationship between nodes, which can be utilized to gain insights into complex datasets. Even though the multi-layered graph is more comprehensive than the single-layer graph, a question that naturally arises is how to fuse the information effectively. Most existing work in the literature focuses on community detection, and an important class of approaches tackles this problem through joint factorization of the multiple graph adjacency matrices to infer embeddings (Tang *et al.*, 2009; Dong *et al.*, 2012). In (Gligorijević *et al.*, 2016), the symmetric non-negative matrix tri-factorization algorithm is utilized in order to factorize the adjacencies into non-negative matrices, including a shared cluster indicator matrix. Other alternative approaches include subgraph pattern mining (Zeng *et al.*, 2006; Boden *et al.*, 2012) and information-theoretic optimization based on Minimum Description Length (Papalexakis *et al.*, 2013). A comprehensive survey studying the algorithms and datasets on this topic can be found in (Kim and Lee, 2015). A unified optimization framework is developed in (Li *et al.*, 2018) to model within-layer connections and cross-layer connections simultaneously to generate node embeddings for interdependent networks. Recently, Song and Thiagarajan (Song and Thiagarajan, 2018) proposed to generalize the DeepWalk algorithm to the case of multi-layered graphs through optimization with proxy clustering costs and showed the resulting embeddings produce state-of-the-art results.

Chapter 3

GRAPH ADVERSARIAL ATTACK AND DEFENSE

Representation learning methods, in particular deep learning, have produced state-of-the-art results in image analysis, language modeling, and more recently with graph-structured data (Torng and Altman, 2019). In particular, graph neural networks (GNNs) (Kipf and Welling, 2017; Hamilton *et al.*, 2017) have gained prominence due to their ability to effectively leverage the inherent structure to solve challenging tasks, including node classification, link prediction, and graph classification (Wu *et al.*, 2020).

Despite their widespread use, GNNs are known to be vulnerable to various adversarial attacks, similar to standard deep models. In other words, a small imperceptible perturbation intentionally designed in the graph structure can lead to non-trivial performance degradation as seen in (Zügner *et al.*, 2018). This is shown in Figure 3.1. This limits their application to high-risk and safety-critical domains. For example, the popular graph convolutional networks (GCN), which rely on aggregating message passes from a node's neighborhood, are not immune to poisoning attacks, wherein an attacker adds fictitious edges to the graph before the model is trained.

Though there exists a vast literature on adversarial attacks on images (Goodfellow *et al.*, 2014; Szegedy *et al.*, 2013) and their countermeasures (Ren *et al.*, 2020; Chakraborty *et al.*, 2018), designing attack strategies for graphs is a more recent topic of research. In general, designing graph attacks poses several challenges: (i) the adversarial search space is discrete; (ii) nodes in the graphs are non-i.i.d., and (iii) lack of effective metrics to measure structural perturbations. Following the progress in graph adversarial attacks, designing defense mechanisms or building robust variants

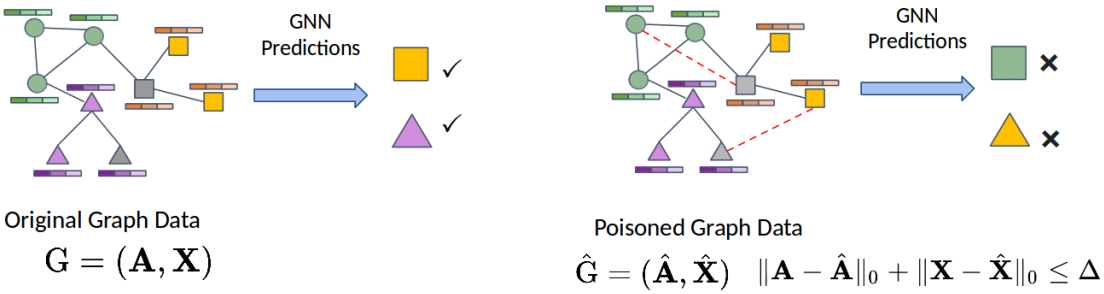


Figure 3.1: A Small Imperceptible Perturbation by the Adversary Fools the GNN Model to Make Wrong Predictions.

of GNNs have become critical (Zhu *et al.*, 2019).

In this chapter, we propose a new approach `UM-GNN` aimed at improving the robustness of GNN models, particularly against challenging poisoning attacks to the graph structure. Our approach jointly trains a standard GNN model (implemented using GCN) and a surrogate predictor, which accesses only the features, using a novel uncertainty matching strategy. The surrogate demonstrates significantly improved robustness to challenging attacks through a systematic knowledge transfer from the GNN model. The key contributions of this work are summarized as follows:

- A novel architecture for semi-supervised learning, `UM-GNN`, that can be built upon any existing GNN model and is immune to evasion attacks by design;
- An uncertainty matching-based knowledge transfer strategy for achieving robustness to structural perturbations;
- Across a suite of global poisoning attacks, `UM-GNN` consistently outperforms existing methods, including the recent Robust GCN (Zhu *et al.*, 2019);
- `UM-GNN` achieves significantly lower misclassification rate ($> 50\%$ improvement) against targeted attacks.

3.1 Problem Setup

In this work, we are interested in building graph neural networks robust to adversarial attacks on the graph structure. We represent an unweighted graph using the tuple $\mathrm{G} = (\mathcal{V}, \mathcal{E})$, where each node v_i may be endowed with a D-dimensional node attribute vector $\mathbf{x}_i \in \mathbb{R}^D$. Alternately the graph data may be represented using an adjacency matrix $\mathbf{A}$ and the node attribute matrix $\mathbf{X}$. We focus on a transductive learning setting as seen in definition 1.3.1, where the goal is to perform node classification. In particular, we assume that we have access to labels for a subset of nodes $\mathcal{V}_L \subset \mathcal{V}$ and we need to predict the labels for the remaining nodes ($v \in \mathcal{V} \setminus \mathcal{V}_L$) in $mathsfG$. Each node v_i is associated with a label $y_i \in \mathcal{Y} = [1, \cdots, C]$.

While a variety of approaches currently exist to solve this semi-supervised learning problem, we restrict our study to the recently successful solutions based on graph neural networks (GNNs). Different flavors of GNNs are reviewed in section 2.3. One of the popular technique is the standard graph convolutional network (GCN) (Kipf and Welling, 2017), which combines message function (equation 2.4) and update function (equation 2.5) in to one as follows

$$\mathbf{h}_i = \sigma\left(\sum_{j \in \mathcal{N}_i} c_{ij} \mathbf{h}_j \mathbf{\Theta}\right) \tag{3.1}$$

Here, the message computation is scaled by $c_{ij} = \frac{1}{\sqrt{d_i d_j}}$, a symmetric normalization constant.

As mentioned earlier, our goal is to defend against adversarial attacks on the graph structure. Formally, we assume that an adversary induces structural perturbations to the graph, i.e., $\hat{\mathrm{G}} = (\hat{\mathbf{A}}, \mathbf{X})$ such that $\|\mathbf{A} - \hat{\mathbf{A}}\|_0 \leq \Delta$. Here, Δ is used to ensure that the adversarial attack is imperceptible. Note that one can optionally also consider the setting where the features $\mathbf{X}$ are also perturbed. While different classes of attacks currently exist (see Section 3.5), we focus on *poisoning* attacks, wherein the graph is

corrupted even before the predictive model is trained. This is in contrast to *evasion* attacks, which assume that the model is trained on clean data and the perturbations are introduced at a later stage. We consider different popular poisoning attacks from the literature (see Section 3.3) and study the robustness of our newly proposed `UM-GNN` approach.

3.2 Proposed Approach

In this section, we present the proposed approach, Uncertainty Matching-GNN (`UM-GNN`), and provide details on the model training process.

While there exist very few GNN formulations for specifically defending against adversarial attacks, the recent robust GCN (RGCN) approach (Zhu *et al.*, 2019) has been the most effective when compared to standard GCN and GAT models. At its core, RGCN relies on using the *aleatoric* uncertainties in the graph structure to weigh the neighborhood. Since there exists no *a* priori knowledge about the structural uncertainties, in practice, simple priors such as the normal distribution (zero mean, unit variance) are placed on the node features and propagated through the network to estimate uncertainties at the output of each layer. Finally, a modified message passing is utilized, wherein neighboring nodes with low feature variance are emphasized during message computation to produce robust features. Despite its empirical benefits, this approach suffers from three main challenges: (i) the choice of the prior is critical to its success; (ii) since the estimated uncertainties are not calibrated, the fidelity of the uncertainty estimates themselves can be low, thus leading to only marginal improvements over GCN in practice; (iii) the model (*epistemic*) uncertainties are not considered, which can impact the generalization of the inferred parameters to the test nodes. In order to alleviate these challenges, we propose `UM-GNN`, a new GNN formulation that uses an uncertainty matching-based knowledge transfer strategy for

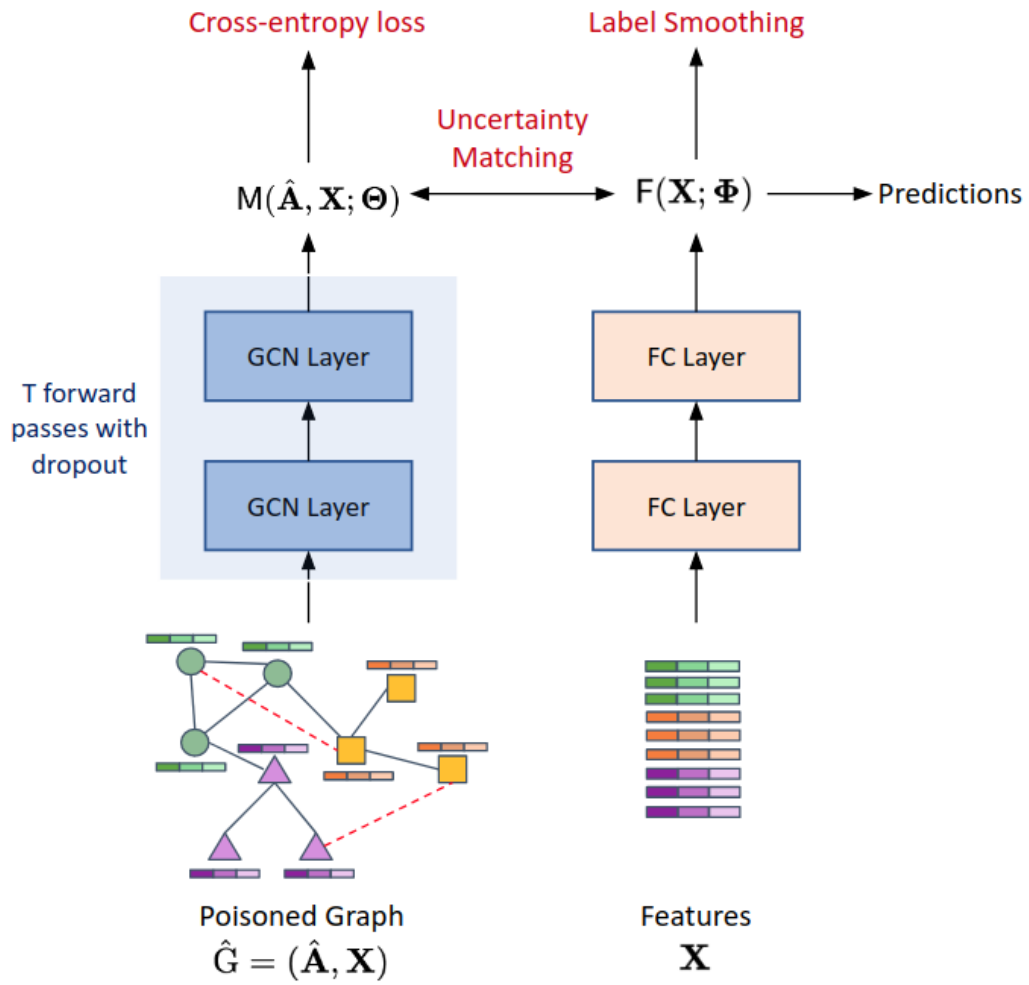


Figure 3.2: An Illustration of the Proposed UM-GNN with GNN Model M and Fully Connected Neural Network F. We Achieve Robustness to Poisoning Attacks Through an Uncertainty Matching Strategy. After the GNN Model M Is Trained, We Use the Surrogate Model F to Make Predictions for the Unlabeled Nodes.

achieving robustness to graph perturbations. In contrast to RGCN, UM-GNN utilizes *epistemic* uncertainties from the GNN and does not require any modifications to the message passing module. As we will show in our empirical studies, our approach provides significant improvements in defending against well-known poisoning attacks.

Figure 3.2 provides an illustration of UM-GNN, which jointly trains a GNN model $\mathsf{M}(\boldsymbol{\Theta})$ and a surrogate model $\mathsf{F}(\boldsymbol{\Phi})$ that is trained solely using the features $\mathbf{X}$ without any knowledge of the graph structure. Here $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ denote the learnable model parameters. Since we expect the graph structure to be potentially corrupted (though severity or type of corruption is unknown), the predictions from the GNN model could be unreliable due to the presence of noisy edges. We reformulate the problem of making M robust into systematically transferring the most reliable knowledge to the surrogate F so that F can make robust predictions. When compared to existing regularization strategies such as GraphMix (Verma *et al.*, 2019), we neither use the (solely) feature-based model F to regularize the training of M nor are the weights shared between the networks. Instead, we build a surrogate predictor that selectively extracts the most reliable information from the "non-robust" M with the hope of being more robust to the noise in the graph structure. Interestingly, by design, the model F does not rely on the graph structure and hence is oblivious to evasion attacks. As shown in Figure 3.2, after training, we only use the surrogate F to obtain the predictions for unlabeled nodes.

3.2.1 Bayesian Uncertainty Estimation

Quantifying the prediction uncertainties in the graph neural network M is at the core of UM-GNN. We propose to utilize Bayesian Neural Networks (BNNs) (Blundell *et al.*, 2015), in particular its scalable variant based on Monte Carlo dropout (Srivastava *et al.*, 2014). In general, dropout variational inference is used to estimate

the epistemic uncertainties: A deep network is trained with dropout. Even at test time, the dropout is used to generate samples from the approximate posterior through Monte Carlo sampling. Interestingly, it was showed in (Gal and Ghahramani, 2016) that the dropout inference minimizes the KL divergence between the approximated distribution and the posterior of a deep Gaussian process. The final prediction can then be obtained by marginalizing over the posterior, using Monte Carlo integration. In our formulation, the node classification task is transductive in nature and does not require test-time inferencing. Hence, we propose to leverage the prediction uncertainties in the training loop itself. More specifically, we obtain the prediction for each node v_i as

$$p(y_i = c; \mathbf{x}_i, \mathbf{A}) = \texttt{Softmax}\left(\frac{1}{T}\sum_{t=1}^{T}\mathsf{M}(\mathbf{x}_i, \mathbf{A}; \tilde{\mathbf{\Theta}})\right)$$

Here, we make T forward passes for $\mathbf{x}_i$ with different masked weights $\tilde{\mathbf{\Theta}}$ (using dropout inference) and compute the final prediction using a sample average. Note, we assume that the predictive model produces logits, i.e., no activation in the final prediction layer, and hence compute the `Softmax` of the average predictions. We then use the *entropy* of the resulting prediction $p(y_i = k; \mathbf{x}_i, \mathbf{A})$ as an estimate of the model uncertainty for node v_i.

$$\begin{aligned}\texttt{Unc}(v_i) &= \texttt{Entropy}\Big(p(y_i = c; \mathbf{x}_i, \mathbf{A})\Big) \\ &= -\sum_{c=1}^{C} p(y_i = c)\log p(y_i = c) \end{aligned} \tag{3.2}$$

3.2.2 Algorithm

We now present the algorithm to train an `UM-GNN` model given a poisoned graph $\hat{\mathrm{G}} = (\hat{\mathbf{A}}, \mathbf{X})$. As described earlier, our architecture is composed of a graph neural network $\mathsf{M}(\mathbf{\Theta})$ and a surrogate model $\mathsf{F}(\mathbf{\Phi})$ that takes only the features $\mathbf{X}$ as input.

While we implement M using graph convolution layers as defined in equation (3.1), it can be replaced using any other message passing strategy, e.g, graph attention layers (Veličković *et al.*, 2018). Given that all datasets we consider in our study contain vector-values defined at the nodes, we implement F as a fully connected network. The optimization problem used to solve for the parameters $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ is given below:

$$\underset{\boldsymbol{\Theta},\boldsymbol{\Phi}}{\text{minimize}}\ L_{ce} + \lambda_m L_m + \lambda_s L_s. \tag{3.3}$$

Here, the first term L_{ce} corresponds to the standard cross-entropy loss over the set of labeled nodes computed using the predictions from the GNN model M.

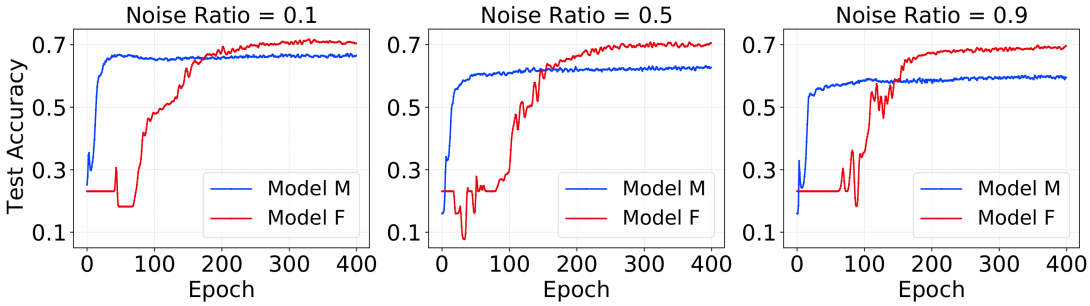
The second term L_m is used to align the predictions between the surrogate F and GNN model M so that the resulting classifiers are consistent. Directly distilling knowledge from the GNN model enables F to actually make meaningful predictions for the nodes, even without accessing the underlying graph structure. However, using a poisoned graph to build M can lead to predictions with high uncertainties. Such noisy examples may lead to unreliable gradients, thus making the knowledge transfer unstable. Hence, we propose to attenuate the influence of samples with high prediction uncertainty. We refer to this process as uncertainty matching and implement it using the KL divergence. However, this can be readily replaced using any general divergence or the Wasserstein metric. Formally,

$$L_m = \sum_{i=1}^{|\mathcal{V}|} \beta_i \, \texttt{KLDiv}(\mathsf{M}(\mathbf{x}_i, \mathbf{A}; \boldsymbol{\Theta}), \mathsf{F}(\mathbf{x}_i; \boldsymbol{\Phi})) \tag{3.4}$$

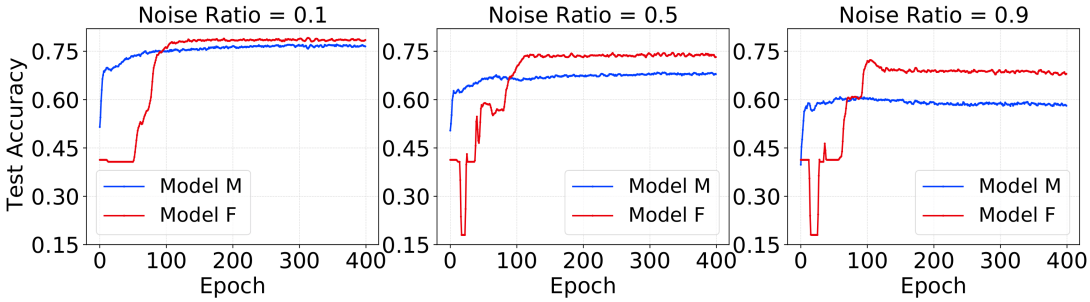
where the weight β_is are computed as

$$\beta_i = \frac{\exp(-\alpha_i)}{\sum_j \exp(-\alpha_j)};\ \text{where } \alpha_i = \log \frac{1}{1 + \texttt{Unc}(v_i)} \tag{3.5}$$

`KLDiv` denotes KL divergence loss measure provided by PyTorch framework. When the prediction uncertainty for a sample is low, it is given higher attention during



(a) Citeseer with Random Attack



(b) Pubmed with DICE Attack

Figure 3.3: Illustration of the Behavior of UM-GNN for Two Datasets Under Varying Types and Levels of Poisoning Attacks. In Each Case, We Show the Test Accuracy Curves Across the Training Epochs From Both the Gnn and Surrogate Models. As the Noise Severity Increases, the Surrogate Model F Demonstrates Improved Robustness.

matching. This loss is evaluated using both labeled and unlabeled nodes since it does not need access to the true labels. Finally, the third term L_s corresponds to a label smoothing regularization that attempts to match the predictions from F to a uniform distribution (KL divergence). This is included to safeguard the surrogate model from being misguided by the graph network when the latter's confidences are not well-calibrated due to the poisoned graph. In all our experiments, we set $\lambda_m = 0.3$ and $\lambda_s = 0.001$. Figure 3.3 illustrates the behavior of UM-GNN for two different datasets under varying levels of poisoning. As the severity of the corruption increases, the surrogate model achieves significantly higher test performance when compared to the

graph-based model M. In cases where no explicit node attributes are available, F may be implemented as a GNN, and the uncertainty matching strategy will still be applicable, and this is part of our future work.

3.3 Poisoning Attacks Used for Evaluation

While there is a broad class of adversarial attacks designed to be applied during the testing phase of the model, we focus on the more challenging poisoning attacks. Poisoning attacks are intended to disrupt the model training by injecting carefully crafted corruptions into the training data. In particular, it is well known that they are highly effective at degrading the performance of GNNs. More importantly, existing robust modeling variants such as RGCN provide only marginal improvements over the standard GNN models when presented with poisoned graphs. Hence, we evaluate the proposed `UM-GNN` using several widely-adopted poisoning attacks. Here, we briefly describe those attacks and provide our implementation details. For each type of attack, we evaluate the model performance by varying the ratio of noisy edges with respect to the total number of existing edges. Any edge that is added or perturbed is referred to as noisy edge in this setup.

Random Attack

This is a purely black-box attack, where the attacker does not know ground truth labels or the model information. More specifically, new edges are randomly introduced in this attack between two nodes that were not previously connected. Though being simple, this attack is known to be effective, particularly at higher noise ratios and sparse graphs. We varied the ratio of noisy edges between 10% and 100% of the total number of edges in the original graph for our experiments.

DICE Attack (Waniek *et al.*, 2018)

This is a gray-box attack where the attacker has information about the node labels but not the model parameters. This attack uses a modularity-based heuristic to Disconnect Internally (nodes from the same community) and Connect Externally (DICE) (nodes from different communities). For a given budget, an attacker randomly deletes edges that connect nodes from the same class; and adds edges between randomly chosen node pairs of samples from different classes. Similar to the random attack, we varied the perturbation ratio between 10% and 100% of the total number of existing edges.

Meta-Gradient Attack

(Mettack) (Zügner and Günnemann, 2019) Mettack is a more challenging gray-box attack where the attacker utilizes the graph structure and labels to construct a surrogate model, which is then utilized to generate the attacks. More specifically, Mettack formulates a bi-level optimization problem of maximizing the classification error on the labeled nodes after optimizing the model parameters on the poisoned graph. In other words, the graph structure is treated as the hyper-parameter to optimize, and this is solved using standard meta-learning strategies. Since the surrogate model is also designed based on GCNs (similar architectures as our predictive model) and trained with the entire graph (transductive setting), this gray-box attack is powerful in practice. Hence we used lower noise ratios for our experiments, i.e., between 1% to 10% of the total existing edges compared to Random and DICE attacks.

Projected-Gradient Attack

(PGD) (Xu *et al.*, 2019) PGD is a first-order topology attack that attempts to determine the minimum edge perturbations in the global structure of the graph, such

Dataset	# Nodes	# Edges	# Features	# Classes
Cora	2708	5278	1433	7
Citeseer	3327	4614	3703	6
Pubmed	19717	44325	500	3

Table 3.1: Summary of the Three Benchmark Citation Datasets Used in Our Experments.

that the generalization can be maximally affected. Since PGD cannot access the true model parameters, we use a surrogate GNN model to generate the attacks. Similar to Mettack, we varied the perturbation ratio between 1% and 10% in this case as well.

Fast Gradient Attack

(FGA) (Chen *et al.*, 2018) FGAs are created based on gradient information in GNNs, and they belong to the category of targeted attacks. The goal of a targeted attack is to mislead the model into classifying a target node incorrectly. In FGA, the attacker adds an edge between node pairs characterized by the largest absolute difference in their gradients. We choose FGA to show the superior performance of `UM-GNN` even against targeted attacks.

The implementations for Mettack, PGD, and FGA were based on the publicly available DeepRobust (Jin *et al.*, 2020) library. Due to the lack of computationally efficient implementations, we could not generate these attacks on large-scale graphs such as Pubmed.

3.4 Empirical Evaluation

In this section, we evaluate the robustness of `UM-GNN` against the graph poisoning methods discussed in the previous section. As mentioned in Section 3.3, non-targeted

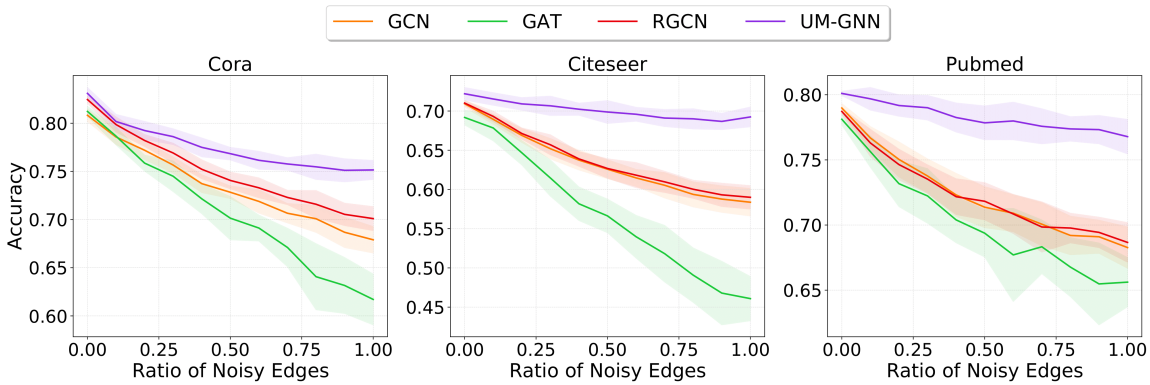


Figure 3.4: *Random Attack*: `UM-GNN` Achieves Robustness to Random Attacks, Providing Over $5-10\%$ Improvements in the Test Accuracy, Even When the Noise Ratio is 1.0.

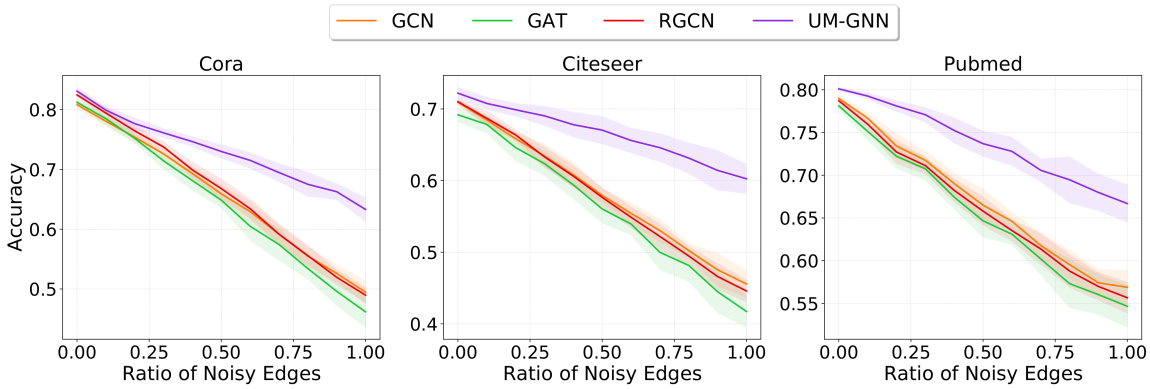


Figure 3.5: *DICE Attack*: For All Datasets, `UM-GNN` Is Consistently More Robust in This Challenging Scenario, Where the Attacker Both Adds and Deletes Edges. The Performance Improvement with `UM-GNN` Is as High as $\approx 15\%$ (Citeseer).

poisoning attacks are far more challenging and pose a more realistic threat to graph-based models.

Datasets

We consider three benchmark citation networks extensively used in similar studies: Cora, Citeseer, and Pubmed (Sen *et al.*, 2008). Nodes represent the documents, and citations among the documents are encoded as undirected edges. We follow the typical transductive node classification setup (Kipf and Welling, 2017; Veličković *et al.*, 2018) while using the standard train, test, and validation splits for our experiments (see Table 3.1).

Baselines

We compare the proposed approach with three important baseline GNN models, which adopt different message-passing formalisms and have been successfully used in semi-supervised node classification tasks. Note that the performance of a feature-only classifier (MLP) which ignores the graph structure, produces trivial performances with the following accuracies: 55.1% for Cora, 46.5% for Citeseer, and 71.4% for Pubmed.
GCN: We use the GCN model, proposed by Kipf & Welling, based on the message passing formulation in equation (3.1).
GAT (Veličković *et al.*, 2018): This model uses a multi-head attention mechanism to learn the hidden representations for each node through a weighted aggregation of features in a closed neighborhood where the weights are trainable.
RGCN (Zhu *et al.*, 2019): This is a recently proposed approach that explicitly enhances the robustness of GCNs. RGCN models node features as distributions as opposed to deterministic vectors in GCN and GAT models. It employs a variance-based attention mechanism to attenuate the influence of neighbors with large variance (potentially corrupted). Following (Zhu *et al.*, 2019), we set hidden dimensions at 16 and assume a diagonal covariance for each node.

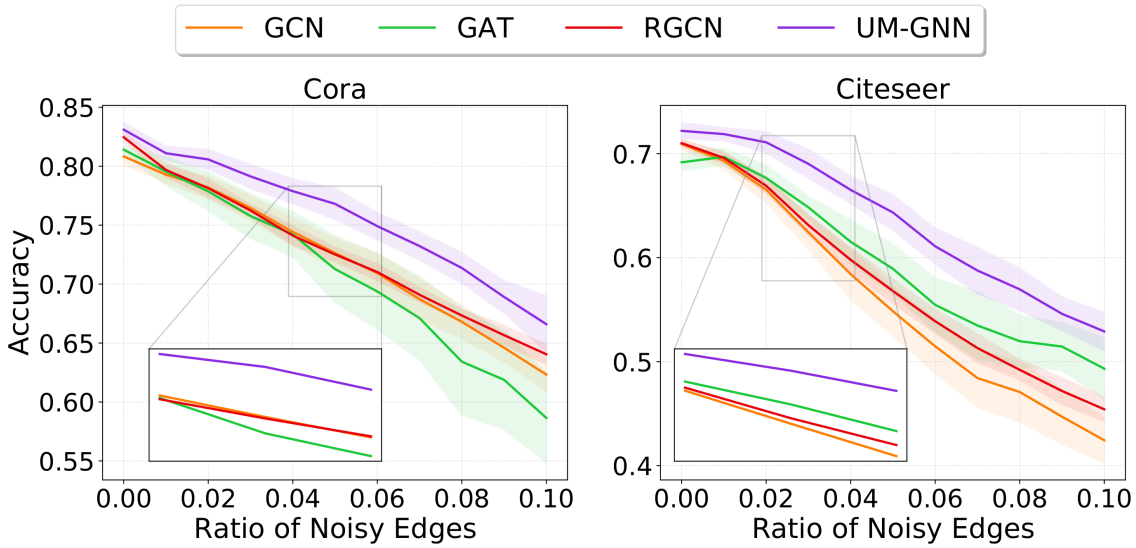


Figure 3.6: *Mettack* - This Gray-box Attack Is Known to Be Highly Effective at Causing Performance Degradation in GNNs. However, `UM-GNN` Consistently Provides $3-5\%$ Improvements in the Test Accuracy over the Baselines.

We set the number of layers (2 layers) and other hyper-parameter settings for all baselines as specified in their original papers. We set the number of hidden neurons to 16 for both GCN and GAT baselines. In addition, we set the number of attention heads to 8 for GAT. We implemented all the baselines and the proposed approach using the Pytorch Deep Graph Library (version 0.5.1) (Wang *et al.*, 2019). In our implementation of `UM-GNN`, the GNN model M was designed as a $2-$layer GCN similar to the baseline, and the surrogate F was a $3-$layer FCN with configuration $32-16-K$, where K is the total number of classes.

3.4.1 Results

We evaluated the classification accuracy on the test nodes for the datasets against each of the attacks under varying perturbation levels. For random and DICE attacks, we varied the ratio of noisy edges to clean edges between 0.1 and 1. Since Mettack

and PGD attacks are more powerful, we used noise ratios in the range $(0.01, 0.1)$. For all the 4 global attacks, we repeated the experiment for 20 random trials (different corruption) for each noise ratio and reported the expected accuracies along with their standard deviations.

(i) Random Attack: The results for random attacks for all three datasets are shown in Figure 3.4. As discussed earlier, RGCN provides only a marginal improvement over the vanilla GCN and GAT. However, `UM-GNN` consistently outperforms the baselines by a large margin even when the ratio of noisy edges to clean edges is high. In addition, `UM-GNN` has the least variance in performance compared to the baselines. In comparison, GAT appears to be the most sensitive to random structural perturbations, and its low performance strongly corroborates with the findings in Zhu *et al.* (2019).

(ii) DICE Attack: In this challenging attack, where the attacker can both delete and add edges, all baseline methods suffer from severe performance degradation when compared to random attacks. Surprisingly, `UM-GNN` is significantly more robust and achieves performance improvements as high as $\approx 15\%$ (Figure 3.5, Citeseer, noise ratio = 1.0). This clearly evidences the ability of `UM-GNN` to infer the true modular structure, even when the graph is poisoned.

(iii) Mettack Attack: Since mettack uses a surrogate model and its parameters to generate attacks, it is one of the more challenging attacks to defend. Nevertheless, `UM-GNN` consistently outperforms all the baselines by a good margin, as illustrated in Figure 3.6. Interestingly, under this attack, both GCN and RGCN perform poorly when compared to the GAT model. However, the large variance makes GAT unreliable in practice, particularly when the attack is severe.

(iv) PGD Attack: This is comparatively the most severe, since the GCN model used to generate the attack has the same architecture as our model

$mathsf{M}$, thus in actuality making it a white-box attack. From Figure 3.7, we observe $1\% - 2\%$ improvements in mean performance over the baselines. More importantly, the lower variance of `UM-GNN` across trials makes it a suitable choice for practical scenarios.

(v) FGA Attack: For this targeted attack, we selected 100 test nodes with correct predictions in a baseline GCN as our targets. Out of the 100 target nodes, 25 nodes were those with the highest margin of classification, 25 nodes were those with the lowest margin, and the remaining 50 were chosen randomly. Further, we set the number of perturbations allowed on each target node to be equal to its degree (so that it is imperceptible). The FGA attack was generated for each target node independently. We checked if the targeted attack was defended successfully or not, i.e., whether the targeted node was classified correctly using the poisoned graph. The overall misclassification rates for the different models are shown in Table 3.2. We find that `UM-GNN` provides dramatic improvements in defending against FGA attacks through its systematic knowledge transfer between the GNN M and the surrogate F. In Figure 3.8, we plot the prediction probabilities for the true class (indicates a model's confidence) for all target nodes obtained using the original and poisoned graphs G and $\hat{\mathrm{G}}$ respectively. As it can be observed, `UM-GNN` improves the confidences considerably for all samples, while the baseline methods demonstrate vulnerability to FGA.

3.5 Related Work

Semi-supervised learning based on graph neural networks (GNNs) enables representation learning using both the graph structure and node features (Wu *et al.*, 2020). While GNNs based on spectral convolutional approaches (Bruna *et al.*, 2013; Defferrard *et al.*, 2016a; Kipf and Welling, 2017) have been widely adopted, there also exists models that implement convolutions directly using spatial neighborhoods (Duvenaud

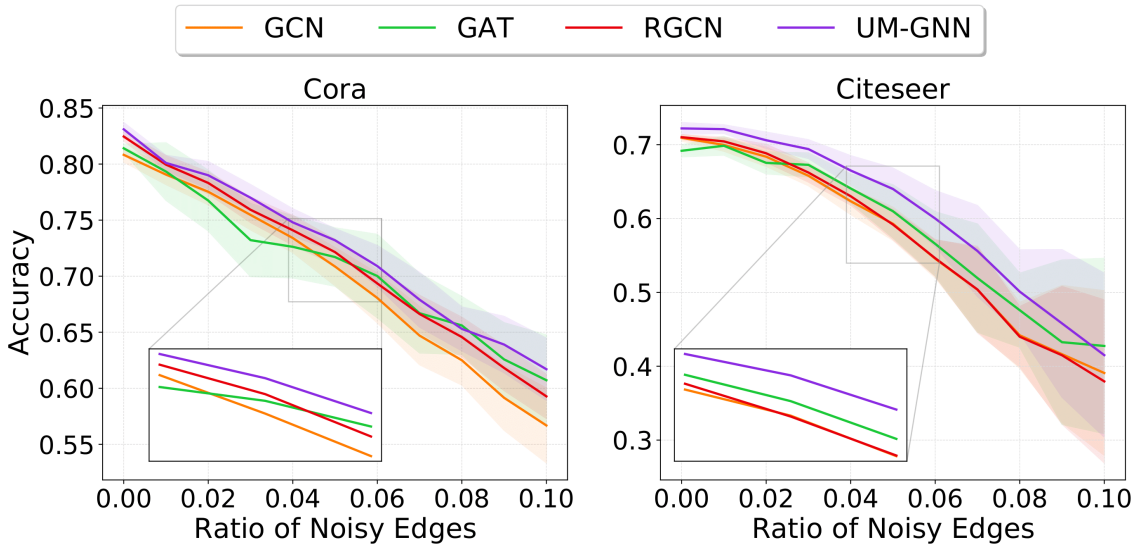


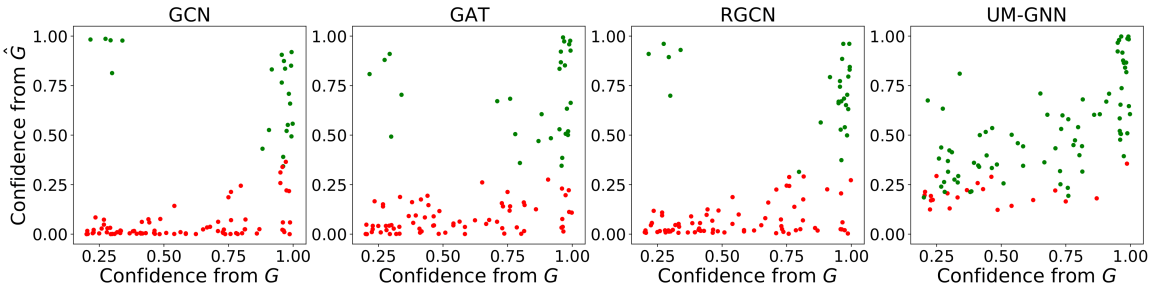
Figure 3.7: *PGD Attack* - This Is Comparatively Very Severe, Since It Uses Gradients from a GCN Model (Same Architecture as M). While the Accuracy Improvements Are Still Non-trivial ($1\% - 2\%$), the More Interesting Observation Is the Reduced Variance of `UM-GNN` Across Trials.

et al., 2015; Atwood and Towsley, 2016; Hamilton *et al.*, 2017).

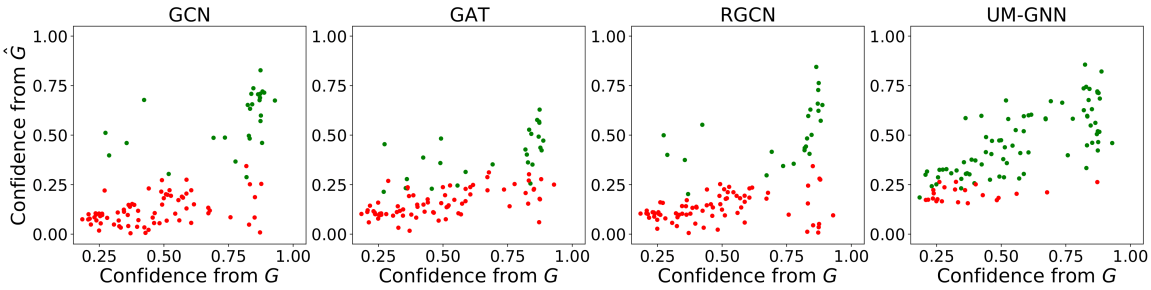
Graph Adversarial Attacks The vulnerability of GNNs to adversarial attacks was first studied in (Zügner *et al.*, 2018). Since then, several graph adversarial attacks have been proposed (Jin *et al.*, 2020; Sun *et al.*, 2018). Adversarial attacks on graphs can be broadly categorized as follows:

(i) *Attacker knowledge*: based on the level of access an attacker has to the model internals, namely white-box (Xu *et al.*, 2019; Wu *et al.*, 2019a), gray-box (Zügner *et al.*, 2018; Zügner and Günnemann, 2019) and black-box attacks (Bojchevski and Günnemann, 2019).

(ii) *Attacker capability*: based on whether the attacker perturbs the graph before Liu *et al.* (2019) or after (Dai *et al.*, 2018) the model is trained.



(a) Cora dataset



(b) Citeseer dataset

Figure 3.8: Results from FGA Attacks on Two Benchmark Datasets - on the X-axis, We Plot the Prediction Probabilities for the True Class Obtained Using GCN on the Clean Graph G. On the Y-axis, We Show the Prediction Probabilities Obtained after the Targeted Attack. Note, for Each Method, We Show the Misclassified Nodes in Red and the Correct Predictions in Green.

(iii) *Attack strategy*: based on whether the attacker corrupts the graph structure or node features. While structural perturbations can be induced by deleting, adding, or re-wiring edges, new nodes could also be injected into the graph (Shanthamallu *et al.*, 2020).

(iv)*attacker's goal*: based on whether the attack is aimed at degrading the model's overall performance (Waniek *et al.*, 2018) or targeting specific nodes either directly or indirectly for their misclassification (Chen *et al.*, 2018).

Model	Cora	Citeseer
GCN	0.78	0.73
GAT	0.71	0.74
RGCN	0.73	0.76
UM-GNN	**0.21**	**0.23**

Table 3.2: Misclassification Rates from 100 Target Nodes with FGA attack. A Lower Value Implies Improved Robustness.

Graph Adversarial Defense As graph adversarial attacks continue to be studied, efforts to design suitable defense strategies have emerged recently. For example, Feng *et al.* adapted the conventional adversarial training approach to the case of graphs in order to make GNNs more robust Goodfellow *et al.* (2014); Feng *et al.* (2019). On the other hand, methods that rely on graph pre-processing have also been proposed. For example, in Wu *et al.* (2019a), edges with low Jaccard similarity between the constituent nodes were removed before training a GNN. Similarly, in Jin *et al.* (2019), explicit graph smoothing was performed by training on a family of graphs to defend against evasion attacks. Entezari *et al.* obtained a low-rank approximation of the given graph and showed that it could defend against specific types of graph attack Zügner *et al.* (2018). Recently, Zhu *et al.* (Zhu *et al.*, 2019) introduced a robust variant of GCN based on a variance-weighted attention mechanism and showed it to be effective against different types of attacks.

3.6 Summary

In this work, we presented UM-GNN an uncertainty matching-based architecture to explicitly enhance the robustness of GNN models. UM-GNN utilizes epistemic un-

certainties from a standard GNN M and does not require any modifications to the message passing module. Consequently, our architecture is agnostic to the choice of GNN to implement M. By design, the surrogate model F does not directly access the graph structure and hence is immune to evasion-style attacks. Our empirical studies evidenced the effectiveness of `UM-GNN` in defending against several graph poisoning attacks, thereby outperforming existing baselines. Furthermore, we showed dramatic improvements in defense against targeted attacks (FGA). Future work includes studying the performance bounds of `UM-GNN` and developing extensions for inductive learning settings.

Chapter 4

GRAMME: GRAPH ATTENTION MODELS FOR MULTI-LAYERED EMBEDDINGS

With the emergence of multi-view datasets in real-world scenarios, commonly represented as *multi-layered* graphs, conventional inferencing tasks have become more challenging. Though multi-layered graphs lead to richer representations, extending solutions from the single-graph case is not straightforward. Consequently, there is a strong need for novel solutions to solve classical problems, such as node classification, in the multi-layered case.

In this chapter, we present a novel approach, *GrAMME* (Graph Attention Models for Multi-layered Embeddings), for constructing multi-layered graph embeddings using attention models. In contrast to the existing literature on community detection, we propose to perform feature learning in an end-to-end fashion with the node classification objective and show that it is superior to employing separate stages of network embedding (e.g., DeepWalk) and classifier design. First, we argue that even in datasets that do not have explicit node attributes, using random features is a highly effective choice. Second, we show that attention models provide a powerful framework for modeling inter-layer dependencies and can easily scale to a large number of layers. To this end, we develop two architectures, *GrAMME-SG* and *GrAMME-Fusion*, that employ deep attention models for semi-supervised learning. While the former approach introduces virtual edges between the layers and constructs a *Supra Graph* to parameterize dependencies, the latter approach builds layer-specific attention models and subsequently obtains consensus representations through fusion for label prediction. Using several benchmark multi-layered graph datasets, we demonstrate the

effectiveness of random features. We show that the proposed approaches significantly outperform state-of-the-art network embedding strategies such as DeepWalk. The main contributions of this work can be summarized as follows:

- For the first time, we develop attention model architectures for multi-layered graphs in semi-supervised learning problems;
- We propose the use of random attributes at nodes of a multi-layered graph for deep feature learning;
- We introduce a weighting mechanism in graph attention to better utilize complementary information from multiple attention heads;
- The *GrAMME-SG* architecture that uses attention models to parameterize virtual edges in a Supra Graph;
- The *GrAMME-Fusion* architecture that performs layer-wise attention modeling and effectively fuses information from different layers;
- We evaluate the proposed approaches on several benchmark datasets and show that they outperform existing network embedding strategies.

4.1 Graph Attention Networks

An attention head in graph attention network (GAT) (Velickovic *et al.*, 2017) parameterizes the local dependencies to determine the most relevant parts of the neighborhood to focus on while computing the features for a node. Details of Attention head and Graph Attention Model are provided in Chapter 2. To quickly summarize, an attention head is comprised of the following steps:

Step 1: Feed-forward layer that transforms each $\mathbf{x}_i \in \mathbb{R}^D$ into $\tilde{\mathbf{x}}_i \in \mathbb{R}^d$.

Step 2: A shared trainable dot-product attention mechanism which learns coefficients for each existing edge in the graph. This is carried out using the transformed attributes of the connected neighbors, $a_{ij} = \langle \mathbf{a}, \tilde{\mathbf{x}}_i || \tilde{\mathbf{x}}_j \rangle$, where $\mathbf{a} \in \mathbb{R}^{2d}$ denotes the parameters of the attention function, and $||$ represents concatenation of features from nodes v_i and v_j respectively.

Step 3: A softmax layer for normalizing the learned attention coefficients across the closed neighborhood, $\alpha_{ij} = \texttt{Softmax}_j(a_{ij}; \forall j \in \mathcal{N}_c(i)), \mathcal{N}_c(i)$ denotes closed neighborhood and $\sum_j \alpha_{ij} = 1$. For simplicity, we represent the normalized attention coefficients for the entire graph as the matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$. Note that this is different from the adjacency matrix $\mathbf{A}$.

Step 4: A linear combiner that performs weighted combination of node features with the learned attention coefficients followed by a non-linearity: $\tilde{\mathbf{z}}_i = \sigma(\mathbf{z}_i)$, where $\mathbf{z}_i = \sum_{j \in \mathcal{N}_c(i)} \alpha_{ij} \tilde{\mathbf{x}}_j$.

The modeling capacity of GATs are improved by using multiple attention heads. Following the *Transformer* architecture (Vaswani *et al.*, 2017), the output latent representations from the different heads can be aggregated using either concatenation or averaging operations.

4.2 Weighted Attention Mechanism

From the discussion of GATs in Chapter 2, it is clear that latent representations from the multiple attention heads can provide complementary information about the node relationships. Hence, it is crucial to utilize that information to produce reliable embeddings for label propagation. When simple concatenation is used, as done in (Velickovic *et al.*, 2017), an attention layer results in features of dimension $K \times d$, where K is the number of attention heads. While this has been effective, one can gain improvements by performing a weighted combination of the attention heads, such

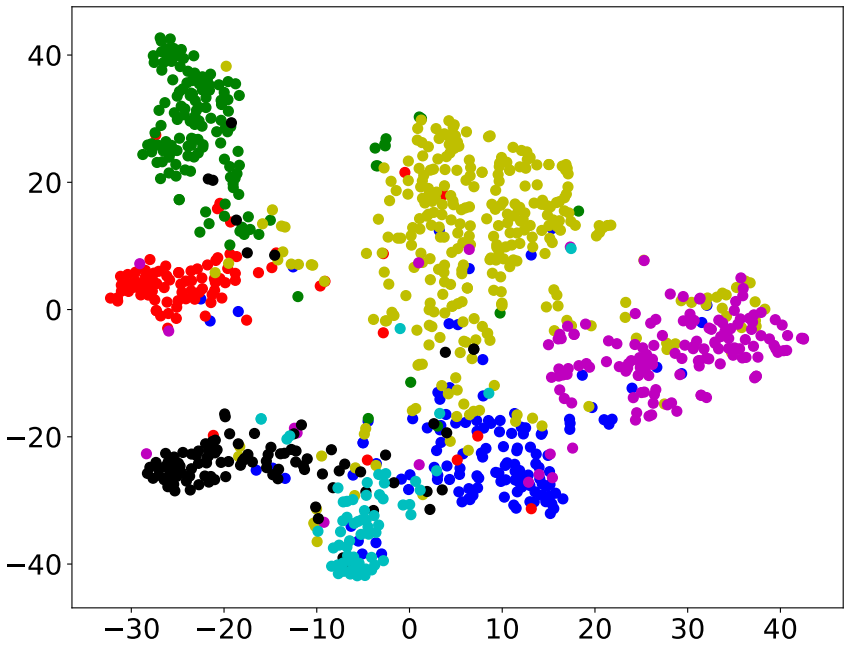


Figure 4.1: 2−D Visualization of the Embeddings for the Single-Layer *Cora* Dataset Obtained Using the Proposed Weighted Attention Mechanism.

that different heads can be assigned varying levels of importance. This is conceptually similar to the *Weighted Transformer* architecture proposed by Ahmed *et al.* (Keskar and Socher, 2017). For a node v_i, denoting the representations from k different heads as $\mathbf{z}_i^1 \cdots \mathbf{z}_i^K$, the proposed weighted attention combines these representations as follows:

$$\hat{\mathbf{z}}_i = \sum_{k=1}^{K} \beta_k \mathbf{z}_i^k, \tag{4.1}$$

where β_k denotes the scaling factor for head k and are trainable during the optimization. Note that the scaling factors are shared across all nodes, and they are constrained to be non-negative. Optionally, one can introduce the constraint $\sum_k \beta_k = 1$ into the formulation. However, we observed that its inclusion did not result in significant performance improvements in our experiments. Given a set of attention heads for a single graph layer, we refer to this weighting mechanism as a *fusion head*.

Interestingly, we find that this modified attention mechanism produces robust embeddings when compared to the graph attention layer proposed in (Velickovic *et al.*, 2017), even with a lesser number of attention heads. For example, let us consider Cora, a single-layered graph dataset containing 2708 nodes (publications) belonging to one of 7 classes. With the regular graph attention model, comprised of two attention layers with 8 heads each, we obtained a test accuracy of 81.5% (140 training nodes). In contrast, our weighted attention, even with just 2 heads, produces state-of-the-art accuracy of 82.7%. Naturally, this leads to a significant reduction in the computational complexity of our architecture, which is more beneficial when dealing with multi-layered graphs. Figure 4.1 illustrates a $2-$D visualization (obtained using t-SNE) of the embeddings from our weighted graph attention model.

4.3 Using Randomized Node Attributes

With graph attention models and other recent graph convolution approaches, it is required to have access to node attributes (or features), which are then used to obtain the latent representations in task-specific objectives. However, in practice, multi-layered graph datasets (even single-layered graphs) are often comprised of only the edge sets, without any additional information. Consequently, in existing graph inferencing approaches (e.g., community detection), it is typical to adopt an unsupervised network embedding strategy. The objective is to ensure that the learned representations preserve the network topology (i.e., neighborhoods). However, such an approach is not optimal for semi-supervised learning tasks since the model parameters can be more effectively tuned using the task-specific objective in an end-to-end fashion.

In order to address this challenge, we propose to employ a randomized initialization strategy for creating node attributes. Interestingly, random initialization has

been highly successful in creating word representations for NLP tasks, and in many scenarios, its performance matches or even surpasses pre-trained word embeddings. With this initialization, the graph attention model can obtain latent representations that maximally support label propagation in the input graph. Unlike fully supervised learning approaches, the embeddings for nodes that belong to the same class can still be vastly different since the attention model fine-tunes the initial embeddings using only the locally connected neighbors. As we will show in our experiments, this simple initialization is effective, and our end-to-end training approach produces superior performance.

4.4 Multi-layer Graph Notation

A multi-layered graph is represented using a set of L inter-dependent graphs $G^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)}), \text{for } l = 1, \ldots, L$, where there exists a node mapping between every pair of layers to indicate which vertices in one graph correspond to vertices in the other. In our setup, we assume $\mathcal{V}^{(l)}$ from all layers contain the same set of nodes with cardinality $|\mathcal{V}| = N$, while the edge sets $\mathcal{E}^{(l)}$ (each of cardinality $M^{(l)}$) are assumed to be different.

4.5 Graph Attention Models for Multi-layered Embeddings (GrAMME)

This section discusses the two proposed approaches for constructing multi-layered graph embeddings in semi-supervised learning problems. As described at the beginning of the chapter, the relationships between nodes are encoded using multiple edge sets in multi-layered graphs. Consequently, while applying attention models for multi-layered graphs, a node v_i in layer l needs to update its hidden state using the knowledge from its neighborhood in that layer and the shared information from other layers. Note that we assume no prior knowledge of the dependency structure

in our proposed approaches and solely rely on attention mechanisms to uncover the structure.

4.5.1 GrAMME-Supra Graph

In this approach, we begin with the initial assumption that information is shared between all layers in a multi-layered graph and use attention models to infer the actual dependencies to improve label propagation performance. More specifically, we introduce virtual edges (also referred to as pillar edges (Kim and Lee, 2015)) between every node in a layer and its counterparts in other layers, resulting in a supra graph, G_{sup}. The block diagonals of the adjacency matrix for G_{sup} contain the individual layers, while the off-diagonal entries indicate the inter-layer connectivities. As illustrated in Figure 4.2, the virtual edges are introduced between nodes with the same ID across layers. This is a popularly adopted strategy in the recent community detection approaches (Song and Thiagarajan, 2018), however, with a difference that the nodes across layers are connected only when they share similar neighborhoods. In contrast, we consider all possible connections for information flow and rely on the attention model to guide the learning process. Note that it is possible that some of the layers can only contain a subset of the nodes. Given a multi-layered graph with L layers, the resulting supra graph G_{sup} is comprised of (at most) $N \times L$ nodes. Furthermore, the number of edges in the supra graph is upper bounded by $(N^2L + NL^2)$, assuming that there are edges between every pair of nodes in every layer, as opposed to N^2L in the original multi-layered graph. The flexibility gained in modeling dependencies comes at the price of increased computational complexity since we need to deal with a much larger graph.

Following this, we generate random features of dimension D at each of the nodes in G_{sup} and build a stacked attention model for feature learning and label prediction.

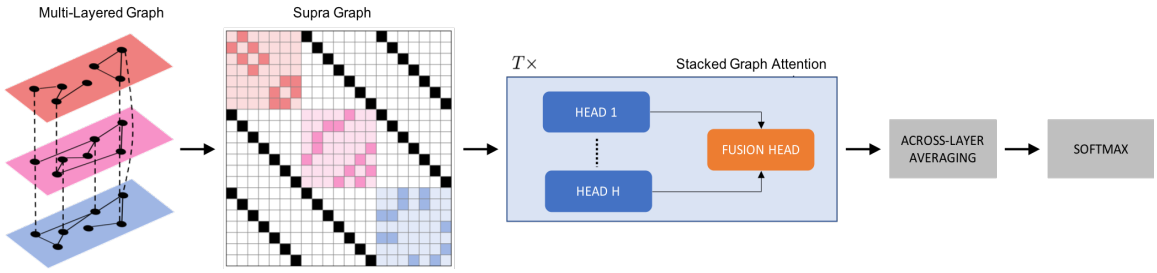


Figure 4.2: *GrAMME-SG* Architecture: Proposed Approach for Obtaining Multi-layered Graph Embeddings with Attention Models Applied to the *Supra Graph*, Constructed by Introducing Virtual Edges Between Layers.

Our architecture comprises T graph attention layers, which in turn contains K attention heads and a fusion head to combine the complementary representations. As discussed earlier, an attention head first performs a linear transformation on the input features and parameterizes the neighborhood dependencies to learn locally consistent features. The neighborhood size for each node can be different, and we also include self-edges while computing the attention weights. Since we are using the supra graph in this case, the attention model also considers nodes from the other layers. This exploits the inter-layer dependencies and produces latent representations that can be influenced by neighbors in the other layers. Following the expression in equation (2.9), the latent feature at a node v_i in layer l can be obtained using an attention head as follows:

$$\mathbf{z}_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i^{(l)} \cup \{v_i^{(1)} \cdots v_i^{(L)}\}} \alpha_{ij}^{(l)} \tilde{\mathbf{x}}_j\right), \tag{4.2}$$

where $\tilde{\mathbf{x}}_j$ denotes the linearly-transformed feature vector for the node v_j. This is repeated with K attention heads with different parameters, and subsequently, a fusion head is used to combine those representations. Note that a fusion head is defined using K scaling factors, denoting the importance for each of the heads. This operation can

be formally stated as follows:

$$\hat{\mathbf{z}}_i^{(l)} = \sum_{k=1}^{K} \beta_k \mathbf{z}_{i^{(l)}}^k. \tag{4.3}$$

Consequently, we obtain latent features of dimension d for each node in G_{sup}, which are then sequentially processed using additional graph attention layers. Since the overall goal is to obtain a single label prediction for each node, there is a need to aggregate features for a node from different layers. For this purpose, we perform an across-layer average pooling and employ a feed-forward layer with softmax activation for the final prediction.

4.5.2 GrAMME-Fusion

While the *GrAMME-SG* approach provides complete flexibility in dealing with dependencies, the complexity of handling large supra graphs is an inherent challenge. Hence, we introduce another architecture, *GrAMME-Fusion*, which builds only layer-wise attention models, and introduces a *supra fusion* layer that exploits inter-layer dependencies using only fusion heads. As described in Section 4.2, a fusion head computes a simple weighted combination and hence is computationally cheap. For simplicity, we assume that the same attention model architecture is used for every layer, although that is not required. This approach is motivated by the observation that attention heads in our feature learning architecture and the different layers in a multi-layered graph both provide complementary views of the same data and hence they can be handled similarly using fusion heads. In contrast, *GrAMME-SG* considers each node in every layer as a separate entity. Figure 4.3 illustrates the *GrAMME-Fusion* architecture.

Initially, each graph layer l is processed using an attention model comprised of T stacked graph attention layers, each of which implements K attention heads and a fusion head to construct layer-specific latent representations. Though the processing

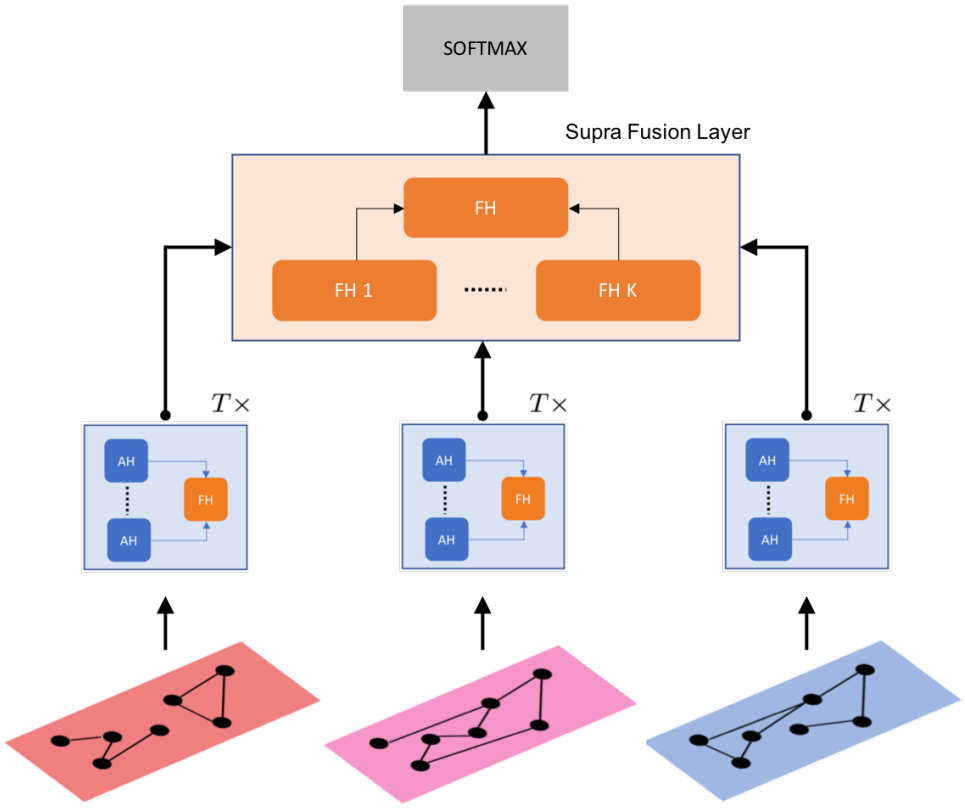


Figure 4.3: *GrAMME-fusion* Architecture: Proposed Approach for Obtaining Multi-layered Graph Embeddings Through Fusion of Representations from Layer-Wise Attention Models.

of the L layers can be parallelized, the computational complexity is dominated by the number of heads K in each model. Next, we construct a *supra fusion* layer, which is designed extensively using fusion heads in order to parameterize the dependencies between layers. In other words, we create H fusion heads with scaling factors $\boldsymbol{\gamma}^{(h)} \in \mathbb{R}^L, \forall h = 1 \cdots H$, in order to combine the representations from the L layer-specific attention models. Note that we use multiple fusion heads to allow different parameterizations for assigning importance to each of the layers. This is conceptually similar to using multiple attention heads. Finally, we use an overall fusion head, with scaling factors $\boldsymbol{\kappa} \in \mathbb{R}^H$, to obtain a consensus representation from the multiple fusion

heads. One can optionally introduce an additional feed-forward layer prior to employing the overall fusion to improve the model capacity. The output from the *supra fusion* layer is used to make the prediction through a fully connected layer with softmax activation. The interplay between the hyper-parameters K (layer-wise attention heads) and H (fusion heads in the *supra fusion* layer) controls the effectiveness and complexity of this approach.

4.6 Empirical Studies and Results

In this section, we evaluate the proposed approaches by performing semi-supervised learning with benchmark multi-layered graph datasets. Our experiments study the behavior of our approaches, with varying amounts of labeled nodes, and cross-validated with different train-test splits. Though the proposed approaches can be utilized for inductive learning, we restrict our experiments to transductive tasks. For each dataset and experiment, we select labeled nodes uniformly at random while fixing the number of labeled nodes. We begin by describing the datasets considered for our study and briefly discussing the baseline techniques based on deep network embeddings.

4.6.1 Datasets

We describe in detail the multi-layered graph datasets used for evaluation. A summary of the datasets can be found in Table 4.1.

(i) Vickers-Chan: The Vickers-Chan (Vickers and Chan, 1981) dataset represents the social structure of students from a school in Victoria, Australia. Each node represents a student studying in 7th grade, and the three graph layers are constructed based on student responses for the following three criteria: (i) whom did they get along within the class?, (ii) who are their best friends in the class?, and (iii) whom do they prefer to work with?. The dataset comprises 29 nodes, and their gender value is used

Table 4.1: Summary of the Datasets Used in Our Empirical Studies.

Dataset	**Type**	**# Nodes**	**# Layers**	**# Total edges**	**# Classes**
Vickers-Chan	Classroom social structure	29	3	740	2
Congress Votes	Bill voting structure among senators	435	4	358,338	2
Leskovec-Ng	Academic collaboration	191	4	1,836	2
Reinnovation	Global innovation index similarities	145	12	18,648	3
Mammography	Mammographic Masses	961	5	1,979,115	2
Balance Scale	Psychological assessment	625	4	312,500	3

as the label in our learning formulation.

(ii) Congress Votes: The Congress votes (Schlimmer, 1987) dataset is obtained from the 1984 United States Congressional Voting Records Database. This includes votes from every congressman from the U.S House of representatives for 4 different bills, which results in a 4-layered graph. The dataset is comprised of 435 nodes, and they are labeled as either democrats or republicans. For every layer, we establish an edge between two nodes in the corresponding layer if those two congressmen voted similarly ("yes" or "no").

(ii) Leskovec-Ng: This dataset (Chen and Hero, 2017) is a temporal collaboration

network of professors Jure Leskovec and Andrew Ng. The 20 year co-authorship information is partitioned into 5-year intervals in order to construct a 4-layered graph. Two researchers are connected by an edge in any layer if they co-authored at least one paper in the considered 5-year interval. Each researcher is labeled as affiliated to either Leskovec's or Ng's group.

(iv) Reinnovation: This dataset describes the Global Innovation Index for 144 countries, which form the nodes of the graph. For each node, the label represents the development level of that corresponding country. There are 3 levels of development, thus representing the 3 classes. Each layer in a graph is constructed based on similarities between countries in different sectors. The sectors include infrastructure, institutions, labor market, financial market, etc. This graph contains 12-layers in total.

(v) Mammography: This dataset contains information about mammographic mass lesions from 961 subjects. We consider different attributes, namely the BI-RADS assessment, subject age, shape, margin, and density of the lesion, in order to construct the different layers of the graph. This data is quite challenging due to the presence of 2 million edges. Conventional network embedding techniques that rely on the sparsity of the graphs can be particularly ineffective in these scenarios. Finally, the lesions are either marked as benign or malignant to define the labels.

(vi) Balance Scale The final dataset that we consider is the UCI Balance scale dataset, which summarizes the results from a psychological experiment. Using 4 different attributes characterizing the subject, namely left weight, the left distance, the right weight, and the right distance, we constructed a $4-$layered graph. Each subject (or node) is classified as having the balance scale tip to the right, tip to the left, or be balanced.

4.6.2 Baselines

Given that the datasets considered do not contain specific node attribute to perform feature learning, the natural approach is to obtain embeddings for each node and subsequently build a classifier model. We compare our proposed architectures with the following state-of-the-art single-layered and multi-layered graph embedding techniques.

DeepWalk: DeepWalk Perozzi *et al.* (2014) is a random-walk-based embedding technique that uses a deep neural network. Random walks on a graph are analogous to sentences in a document, and hence co-occurring nodes are embedded together.

Node2Vec: Node2Vec Grover and Leskovec (2016) is similar to DeepWalk, but it introduces bias in random walks with two additional parameters that trade-off between depth-first and breadth-first walks.

LINE: LINE Tang *et al.* (2015) is similar to DeepWalk but adds information from second hop friends in its random walks. This enables nodes with shared neighborhoods to have similar embeddings.

PMNE: This method Liu *et al.* (2017) uses different merge strategies to combine embeddings from each of the layers in a multi-layered network. We consider the results aggregation strategy since it often outperforms other variants.

MNE: This recent multiplex network embedding Zhang *et al.* (2018a) technique uses a unified network embedding model that generates, for each node, a high-dimensional common embedding and low dimensional embedding for each aspect of the relationship.

4.6.3 Experiment Setup

This section describes the experimental setup in detail for both the baseline methods and the proposed models. We run our experiments in a transductive learning setting. We choose a fixed amount of labeled nodes uniformly at random for each dataset, while the remaining nodes are used for performance evaluation. In order to study the sensitivity of the proposed approaches over varying levels of labeled data availability, we varied the percentage of train nodes from 10% to 30%. We repeated the experiments over 20 independent realizations of train-test splits, and we report the average performance in all cases. The performance of the algorithms was measured using the overall accuracy score.

Since the first three baseline methods (DeepWalk, Node2Vec, LINE) are single-layer graph embedding techniques, we treat each layer in the multi-layered graph data independently and obtain embeddings for the layers separately. Subsequently, we average the embeddings for each node and build a logistic regression classifier to perform label prediction. For DeepWalk and Node2Vec, we set the embedding dimension to 128, the window size to 10, and the number of random walks to 80. For LINE, we fixed the embedding dimension at 100. Among the three variants of PMNE Liu *et al.* (2017), namely network aggregation, Co-analysis, and result aggregation, we report the results only for the result aggregation method, as it often outperforms other variants. The hyper-parameter values for this method were chosen following the original paper. For MNE, a common embedding size of 200 and a layer-specific embedding size of 10 were used.

For both of the proposed approaches, we considered architectures with $T = 2$ attention layers and fixed the input feature dimension $D = 64$. The number of hidden dimensions was fixed at 32. For the GrAMME-SG architecture, we used

$K = 2$ attention heads and a single fusion head. On the other hand, in GrAMME-Fusion, we set $K = 2$ for each layer, and in the supra fusion layer, we used $H = 5$ fusion heads. All networks were trained with the Adam optimizer, with the learning rate fixed at 0.001.

4.6.4 Results

Table 4.2 summarizes the performance of our approaches on the 7 multi-layered graph datasets, along with the baseline results. Figure 4.5 illustrates the convergence characteristics of the proposed GrAMME-Fusion architecture under different training settings for the *Mammography* dataset. As it can be observed, even with the complex graph structure (around 2 million edges), the proposed solutions demonstrate good convergence characteristics.

From the reported results, we make the following observations: In all the datasets, the proposed attention-based approaches consistently outperform the baseline techniques, providing highly robust models even when the training size was fixed at 10%. For example, with the *Vickers-Chan* dataset, both our approaches produce an improvement of over 25% when compared to a weaker baseline such as DeepWalk, and about 14% improvement over the state-of-the-art MNE technique. Even with challenging datasets such as *Reinnovation* and *Mammography* datasets, the proposed approaches achieve improvements of $4\% - 10\%$ over the baseline methods. This clearly demonstrates the effectiveness of our multi-layered graph embedding approaches in scenarios with heterogeneous relationships. Note that *Balance Scale* dataset is the only case where we found the PMNE baseline to be superior to the proposed approaches, however by a minimal margin.

Note that the GrAMME-Fusion has the best performance on almost all the datasets for different train-test splits. Results summarized in Table 4.2 are shown in a different

perspective in Figure 4.4. Here we show the boxplot of the absolute differences between every method and the best performing method for all datasets. A large range in the boxplot indicates that the method is performing badly for some datasets. GrAMME-Fusion has the lowest value and the smallest range.

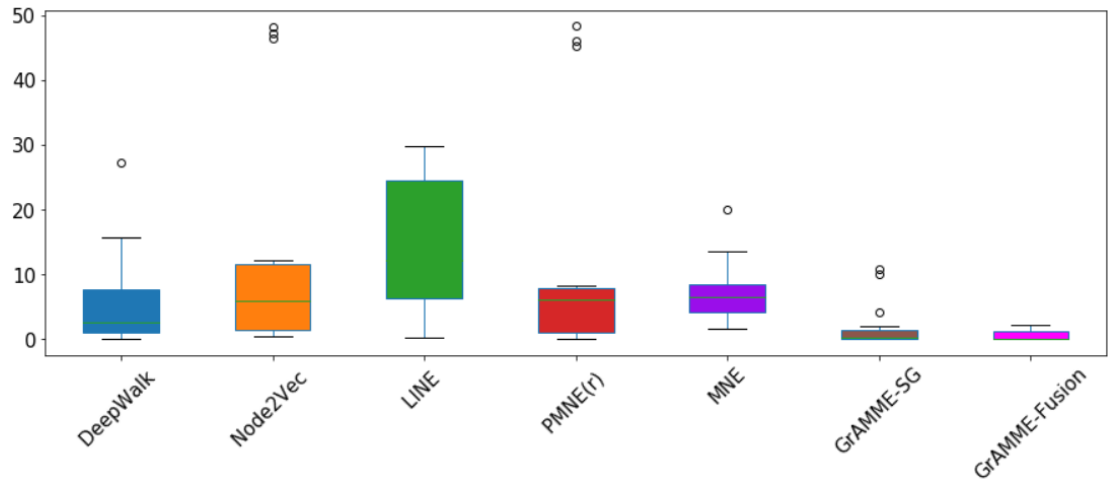
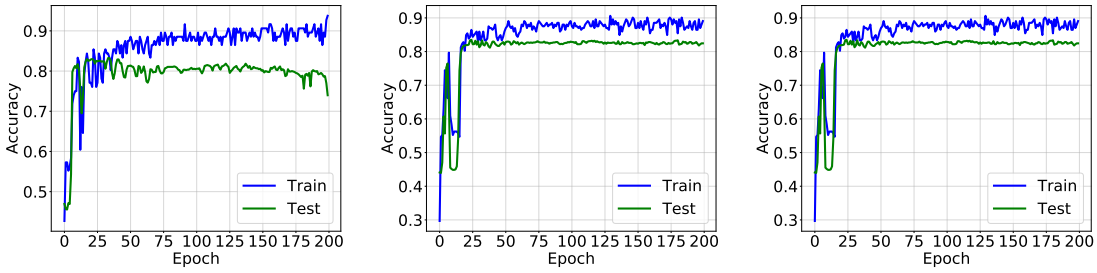


Figure 4.4: Boxplot Showing Absolute Differences of Each Method to the Best Performing Method on Each Dataset for Different Train Test Split. Lower Values Are Better.

Finally, we visualize the multi-layered graph embeddings to qualitatively understand the behavior of the proposed approach. More specifically, we show the $2-$D t-SNE visualizations of the hidden representations for *Congress Votes* and *Mammography* datasets, obtained using GrAMME-Fusion. Figure 4.6 shows that initial random features and the learned representations, wherein the effectiveness of the attention mechanism in revealing the class structure is clearly evident.

4.7 Comparing GrAMME-SG and GrAMME-Fusion

GrAMME-SG operates under the assumption that information is shared between all layers in a multi-layered graph and uses attention models to infer the actual de-

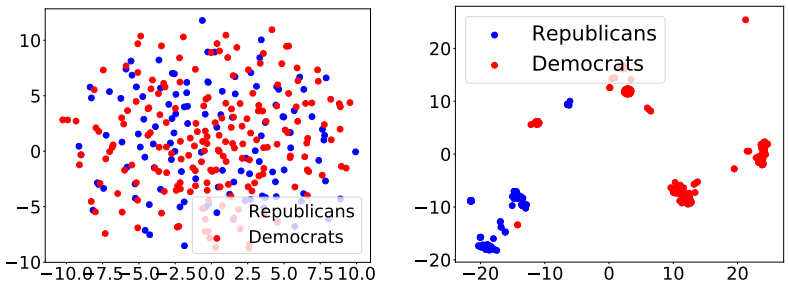


(a) 10% Train Nodes (b) 20% Train Nodes (c) 30% Train Nodes

Figure 4.5: Convergence Characteristics of the Proposed *Gramme-Fusion* Architecture with the Parameters $T = 2$, $H = 1$ and $K = 5$ Respectively.

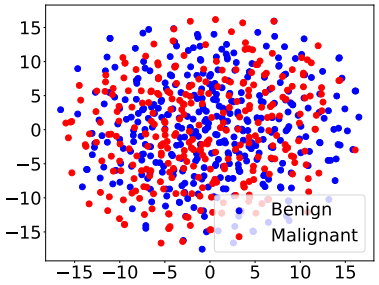
pendencies. GrAMME-Fusion, on the other hand, builds only layer-wise attention models and introduces a *supra fusion* layer that exploits the most relevant inter-layer dependencies using only fusion heads. Though GrAMME-Fusion outperforms GrAMME-SG in most of the datasets considered in our evaluation, we believe this is because GrAMME-SG over-parametrizes inter-layer dependencies and can sometimes produce noisy edges. Consequently, in scenarios where strong dependencies exist between layers, GrAMME-SG will be more appropriate. For example, with the re-innovation dataset, different layers represent each country's performance in diverse sectors such as infrastructure, institutions, labor market, etc. A country with excellent infrastructure and is financially stable can be expected to have a superior labor market and high-quality institutions. As our experiments results show, in that case, GrAMME-SG produces the best performance.

We now present an analysis of the time complexity for the proposed methods. At their core, an attention layer that takes in a single-layered graph with D dimensional attributes and produces d dimensional embeddings incurs a computational complexity of $\mathcal{O}(NDd + Md)$. The first term corresponds to the linear feed-forward layer, while the second term accounts for the attention computation. Note, in cases where

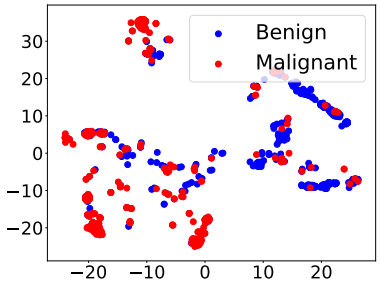


(a) Congress Votes – Initial

(b) Congress Votes – Final



(c) Mammography – Initial



(d) Mammography – Final

Figure 4.6: $2D$ Visualization of the Embeddings, for Two Different Datasets, Obtained Using the *Gramme-Fusion* Architecture with Parameters $T = 2$, $K = 1$ and $H = 5$ Respectively. We Also Show the Initial Randomized Features for Reference.

the graph is densely connected, the second term can dominate the complexity. For GrAMME-SG, we explicitly construct a supra-graph consisting a total of NL nodes and $NL^2 + \sum_{l=1}^{L} M^{(l)}$ edges. Here, the first term corresponds to virtual pillar edges introduced across layers, while the second terms is the sum of layer-specific edges. The computational complexity of an attention head in GrAMME-SG can hence be expressed as $\mathcal{O}(NLDd + NL^2d + d\sum_{l=1}^{L} M^{(l)})$. Consequently, in this case, the number of nodes N plays a more dominant role when compared to the single-layered case. The flexibility gained in modeling dependencies across layers comes at the price of increased computational complexity since we need to deal with a much larger graph.

On the other hand, GrAMME-Fusion is computationally efficient since it employs multiple fusion heads (supra fusion layer), while simplifying the layer-wise attention

models. The complexity of an attention head in this case is given as $\mathcal{O}(NDd + \bar{M}d)$, where $\bar{M}$ indicates $max(M^{(l)})$. Note that the time complexity is similar to that of the single-layered graph. Interestingly, with the GrAMME-Fusion architecture, increasing the number of attention heads K does not lead to significant performance improvements, demonstrating the effectiveness of supra fusion layers. Note that an attention head is computationally expensive when compared to a fusion head in the supra-fusion layer. Consequently, restricting $K = 1$ and increasing the number of fusion heads H leads to a graceful increase in the overall complexity.

More importantly, compared to classical network embedding techniques, this approach is scalable to large-scale graphs, both in terms of N and L, since we do not have to deal with the explicit decomposition of Laplacian matrices. Finally, similar to existing attention models, both the proposed approaches incur $\mathcal{O}(1)$ sequential computations and hence can be entirely parallelized.

4.8 Summary

This chapter introduced two novel architectures, GrAMME-SG and GrAMME-Fusion, for semi-supervised node classification with multi-layered graph data. Our architectures utilize randomized node attributes and effectively fuse information from both within-layer and across-layer connectivities through a weighted attention mechanism. While GrAMME-SG provides complete flexibility by allowing virtual edges between all layers, GrAMME-Fusion exploits inter-layer dependencies using fusion heads, operating on layer-wise hidden representations. Experimental results show that our models consistently outperform existing node embedding techniques. As part of future work, the proposed solution can be naturally extended to the cases of multimodal networks and interdependent networks. Furthermore, studying the effectiveness of simple and scalable attention models in other challenging graph inferencing

tasks such as multi-layered link prediction and influential node selection remains a significant open problem.

Table 4.2: Semi-Supervised Learning Performance of the Proposed Multi-layered Attention Architectures on the Benchmark Datasets. The Results Reported Were Obtained by Averaging 20 Independent Realizations.

% Nodes	Baselines					GrAMME	GrAMME
(Train)	DeepWalk	Node2Vec	LINE	PMNE(r)	MNE	SG	Fusion
Vickers-Chan Dataset							
10%	72	51.07	76.60	50.87	85.76	98.94	**99.21**
20%	83.55	51.97	87.06	53.28	88.37	98.94	**99.21**
30%	89.97	52.88	89.97	53.88	91.72	98.94	**99.21**
Congress Votes Dataset							
10%	98.96	98.46	97.03	98.88	95.62	**100**	**100**
20%	99.69	99.50	98.80	99.75	97.56	**100**	**100**
30%	99.99	99.55	99.70	99.77	98.37	**100**	**100**
Leskovec-Ng Dataset							
10%	91.16	81.76	68.54	85.58	73.34	91.56	**93.32**
20%	96.35	85.41	77.39	89.71	85.79	96.25	**97.62**
30%	98.31	86.99	83.58	91.35	89.90	98.30	**98.73**
Reinnovation Dataset							
10%	72.02	72.18	51.98	70.76	72.51	**76.42**	75.28
20%	73.13	74.04	55.21	73.45	75.40	**80.72**	79
30%	76.02	76.13	60.13	75.29	74.72	**83.16**	80.95
Mammography Dataset							
10%	75.72	76.38	76.39	76.48	75.13	82.27	**82.63**
20%	73.99	77.41	75.40	76.91	76.72	83.01	**83.28**
30%	74.13	77.82	76.16	75.51	77.59	83.06	**83.75**
CKM (Social) Dataset							
10%	97.31	95.70	90.88	97.42	92.86	96.65	**98.66**
20%	98.12	97.92	94.35	98.20	95.27	**99.14**	98.91
30%	99.08	98.34	96.32	98.34	96.83	99.19	**99.68**
Balance Scale Dataset							
10%	81.07	80.58	54.08	**81.85**	77.71	77.67	80.15
20%	86.15	86.22	58.95	**88.74**	80.31	78.67	86.58
30%	87.27	88.61	64.44	**89.87**	83.34	79.10	88.72

Chapter 5

GNN APPLICATION TO HEALTHCARE DATA: HUMAN BRAIN CONNECTOME

Inspired by the effectiveness of deep learning methods in vision, speech, and language processing, there is growing interest in extending those techniques to high-impact application domains such as healthcare. While much of demonstrated success has been on dealing with clinical images/volumes and textual reports, more recent efforts have focused on challenging data sources, including multi-modal health records, knowledge graphs, etc. These efforts have relied on generalizing the foundational formalisms such as convolutional neural nets to arbitrarily structured data. For example, graph neural networks (GNNs) (Wu *et al.*, 2019b) are known to have strong expressive capability with graph-structured data. They have been found to be highly effective with population graphs in clinical diagnosis, e.g., autism (Anirudh and Thiagarajan, 2019).

In this work, we study the use of deep neural networks to analyze brain connectomes, a comprehensive map of neural connections in the human brain. Recently, the Human Connectome Project (HCP) (Van Essen *et al.*, 2013) has made significant strides in producing elaborate structural and functional connectivity of neural pathways in the brain. By leveraging the corresponding neuroimaging data of a human subject, it is now possible to construct a connectivity matrix encoding the neural connections (e.g., number of fibers) between brain regions. Commonly referred to as the *structural connectome*, this connectivity matrix is known to contain signatures relevant to population characteristics such as gender and age Ingalhalikar *et al.* (2014), and more importantly, information pertinent to determining the volumes of

different brain regions. For the first time, we propose to utilize deep neural networks to process the structural connectome directly and test the hypotheses on the predictability of population characteristics and region-specific volumes. This study is a critical first step towards building predictive models from functional connectomes that can reliably predict behavioral traits and cognitive states.

Though it seems natural to interpret the structural connectome for each subject as a graph and employ off-the-shelf graph processing tools, there is a fundamental difference between the connectome and conventional graph datasets such as social networks. With information diffusion-style networks that we typically deal with in practice, an edge between two nodes indicates the likelihood of information exchange between those two nodes. However, even if two nodes are not directly connected by an edge, information exchange can still happen through diffusion from related nodes. In contrast, connectomes encode neural connections as a relational structure and do not allow information diffusion (Zhang *et al.*, 2018b). In other words, a missing edge indicating the absence of connections between two brain regions is a neurological pattern. Consequently, we require network architectures to directly utilize the connectome as a structural prior and infer mappings to the target variables. Similar ideas have been explored in the context of relational reasoning and causal processes (Santoro *et al.*, 2017; Kilbertus *et al.*, 2017).

We develop a relational graph neural network (RGNN) designed to incorporate the relational structure of the neural connections and infer effective latent representations for the connectomes. First, we utilize the edge attribute (e.g., number of fibers) as the input edge feature and construct node representations through constrained message passing with learnable weights. More specifically, we design two network layers that explicitly correspond to the edges and nodes and constrain the message passing between the two layers using the connectome. In other words, the represen-

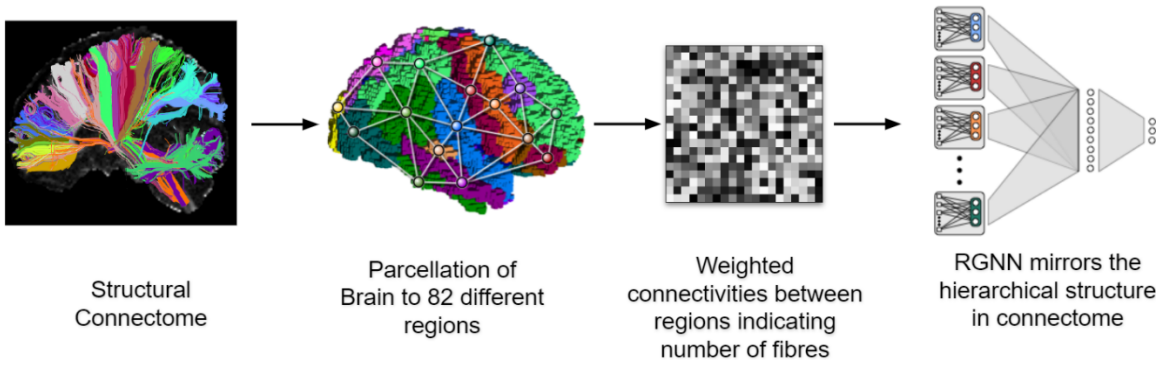


Figure 5.1: An Overview of the Proposed Approach. The T1 Scans Are Parcellated into 84 Different Regions in the Brain, on Which, Tractography Is Performed to Compute the Weighted Matrix Representing the Structural Connectome. Our RGNN Model Is Used to Process the Connectome for Prediction Tasks.

tation for each node can only depend on the edges involving that node. Interestingly, this architecture is effective for even densely connected graphs. Another interesting feature of our approach is that we can produce interpretable explanations (e.g., Shap analysis (Lundberg and Lee, 2017)) both at the coarse-grained node level and the fine-grained edge level simultaneously.

From our experiments, we find that state-of-the-art approaches, including GNN and random walk-based embedding techniques such as *DeepWalk*, are very ineffective at recovering the region-specific volumes. Surprisingly, even a simple fully connected network on the edge attributes outperforms the graph network baselines. In comparison, the proposed RGNN accurately estimates the region-specific volumes and summary quantities such as total gray matter and total white matter. Furthermore, our prediction results on gender prediction strongly corroborate with the findings in the neuroscience community (Ingalhalikar *et al.*, 2014). This evidences the effectiveness of RGNN as a learning strategy for connectomes, and we believe this can be

suitable for studying functional connectomes.

5.1 Human Connectome Data

We used data from the Human Connectome Project - Young Adult dataset[1], which included about 900 subjects. In order to generate the structural connectome for each subject, we performed the following steps: First, we segmented the T1 images into five tissue types (MRtrix[2] command `5ttgen`) (Smith *et al.*, 2012). Then we used the diffusion MRI data along with the tissue segmented volume to produce tissue-specific response functions (MRtrix command `dwi2response`) (Jeurissen *et al.*, 2014). Subsequently, this response was used to construct fiber orientation distributions using the spherical deconvolution (MRtix command `dwi2fod`) (Tournier *et al.*, 2004) and tractography was carried out using the iFOD2 technique (MRtrix command `tckgen`) (Tournier *et al.*, 2010). Finally, we employed the SIFT2 method to compute streamline weights (MRtrix command `tcksift2`) (Smith *et al.*, 2015), and the weighted streamline counts were used along with the Desikan parcellation to compute the connectome matrix (MRtrix command `tck2connectome`) (Desikan *et al.*, 2006).

5.2 Approach

We describe the RGNN architecture that leverages the relational structure of connectomes. Though the focus of this work is on structural connectomes, our approach applies to functional connectomes as well. Our approach uses constrained message passing kernels, with learnable parameters, on the connectome edges to obtain node (region) representations. To this end, we design two neural network layers: the first layer representing the neuron connections takes as input the vectorized set of edge weights of size $N \times N$, where N is the total number of regions in the connectome.

[1]https://www.humanconnectome.org/study/hcp-young-adult

[2]MRtrix version 3.0, https://www.mrtrix.org

The second layer corresponds to the N nodes (or regions). The relational structure from the connectome is directly used to construct messaging passing kernels between the two layers. In other words, the feature for each node in the second layer should depend only on the edges involving that node, i.e.,

$$h_i = \sigma(\sum_{j=1}^{n} W_{ij}\theta_j) \tag{5.1}$$

where h_i is the learned representation for node v_i, W_{ij} is the connectome edge weight between node v_i and v_j, $\boldsymbol{\theta}_i \in \mathbb{R}^N = [\theta_1, \theta_2 \cdots \theta_N]^T$ are the learnable parameters of the kernel associated with node v_i, and σ is a non-linearity function (e.g. ReLU).

From another perspective, we pool all information from the edges involving a node to constructing its representation. Furthermore, we allow the use of multiple kernels for the same node, akin to multiple heads in attention models, and we refer to this as the pooling size P. In the case of multiple kernels, we concatenate the P representations for each node, thus producing a node-level representation of dimensions P and graph level feature of effective size $N \times P$.

$$\mathbf{h}_i = \|_{p=1}^{P} h_i^{(p)} \quad \text{and} \quad \mathbf{H} = \|_{i=1}^{N} \mathbf{h}_i$$

where $\|$ indicates concatenation. Though the constrained message-passing kernel can be implemented differently, we adopt a simple strategy where a fully connected layer is used to produce a weighted sum of edge connectivities, and the network parameters are optimized using backpropagation. Note, we use different kernels for each of the nodes, and their weights are not shared. Finally, we pass the node representations $\mathbf{h}_i$ through a classifier or a regressor implemented using additional fully connected layers. All volume estimation models are trained with the smooth L1 loss, while the age/gender prediction uses the cross-entropy loss. Figure 5.2 shows a representative architecture of RGNN.

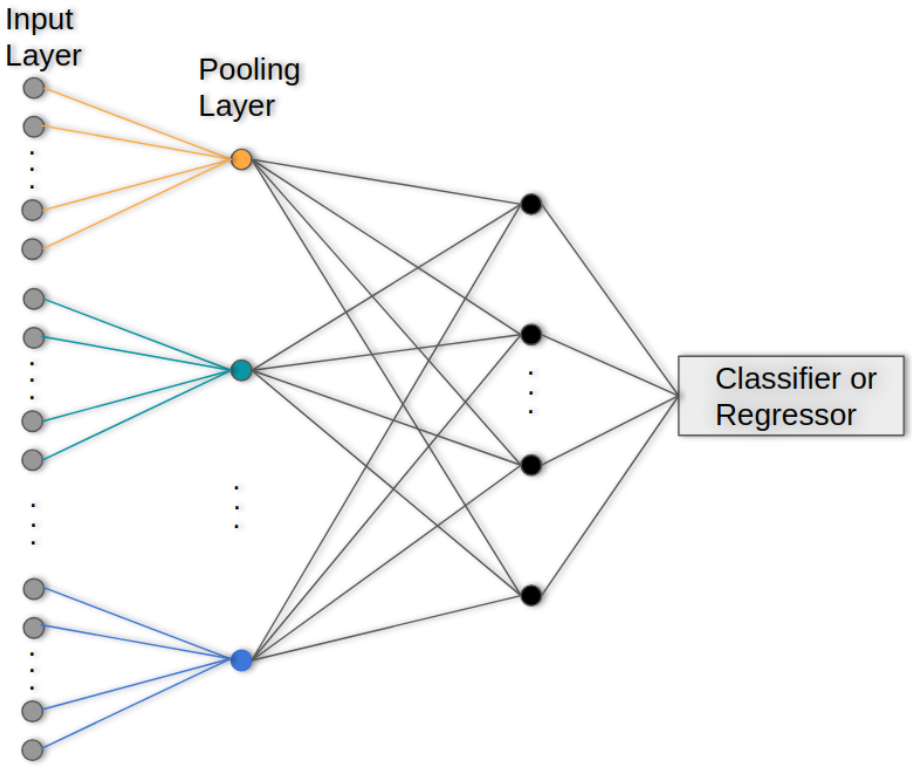


Figure 5.2: An Overview of Proposed Approach RGNN. The Information Is Pooled from the Edges to a Single Node (Region in the Brain).

5.3 Empirical Evaluation

We validate the suitability of RGNN to analyze connectomes by attempting to estimate the region-specific volumes in different parts of the brain and predict meta-information, such as age and gender, directly based on the neural connections. This empirical study was motivated by existing works in the neuroscience literature, which have hypothesized that specific signatures that can predict the volume statistics and gender/age information are encoded in the connectome. Consequently, we expect an appropriate neural network architecture to adhere to these hypotheses.

In all experiments with RGNN, we set the pooling size to 5 and the hidden layer size at 16. For comparison, we consider the following popularly adopted solutions:

Table 5.1: Estimating Region-specific Volumes Using the Structural Connectome. For Each Case, We Report the R^2 / Pearson Correlation Coefficient Metrics.

Region	**DeepWalk**	**GCN**	**FCN**	**RGNN**
Total gray matter	0.4444 / 0.675	-0.0116 / 0.1829	0.6515 / 0.8197	**0.7219 / 0.8577**
Total white matter	0.3753 / 0.62	-0.0132 / 0.1459	0.7011 / 0.8559	**0.8029 / 0.9078**
Left hemisphere cortical white matter	0.3849 / 0.6281	-0.0133 / 0.1561	0.7022 / 0.8572	**0.8027 / 0.9082**
Right hemisphere cortical white matter	0.3731 / 0.6173	-0.0132 / 0.1351	0.6967 / 0.8534	**0.7975 / 0.9047**
Left hemisphere cortical gray matter	0.4342 / 0.6688	-0.0102 / 0.1092	0.6231 / 0.8017	**0.6987 / 0.8449**
Right hemisphere cortical gray matter	0.4525 / 0.6817	-0.0114 / 0.2206	0.6491 / 0.8159	**0.7172 / 0.8580**
Intracranial	0.3627 / 0.6084	-0.0175 / 0.1469	0.5987 / 0.7920	**0.6652 / 0.8239**

a) FCN: In this simple baseline, we ignore the network structure and vectorize the connectivity matrix for each subject to produce a feature vector of 7056 dimensions (84×84). Subsequently, we build a fully connected network with a single hidden layer of 16 units and ReLU non-linearity.

b) DeepWalk Perozzi *et al.* (2014): For this random-walk baseline, we first build a supra graph, whose adjacency is constructed by stacking the connectomes as block diagonals. We run the DeepWalk algorithm on this supra graph and extract 64-d features for each node. Note, for better results, we sparsified each connectome by retaining only the top 75% values. Finally, the representation for each connectome is obtained by concatenating all its node embeddings, and the prediction is carried out

using XGBoost.

c) Message Passing GCN Kipf and Welling (2016): By treating each connectome as a graph, we build a 2-layer GCN model based on weighted message passing. Since the nodes do not have explicit attributes, the message passing with edge weights does not produce meaningful features. Hence, the performance of GCN is poor in regression experiments compared to DeepWalk. We experimented with different initializations for the node attributes, including a constant attribute at each node and the eigenfunctions. Finally, we concatenate the transformed node features to obtain the connectome representation and utilize a fully connected layer for the actual prediction.

Volume Estimation: In this experiment, we considered the prediction of volumes in the following regions: cortical white matter and cortical gray matter volumes in the left and right hemispheres of the brain, and finally, the summary total gray matter and total white matter volumes. Table 5.1 reports the performance of the volume estimation experiment. All the results reported were obtained using an $80\% - 20\%$ train-test split of 900 subjects and by aggregating the performance from 20 random trials. We use the R^2 statistic and the Pearson correlation coefficient as performance metrics. As it can be observed, RGNN consistently outperforms all baseline methods by a significant margin, and more importantly, the GCN based on a diffusion-network assumption fails. In comparison, even the naïve FCN baseline produces meaningful estimates. This clearly demonstrates the need for specific architectures that can leverage the relational structure of connectomes.

Predicting Age/Gender: In this experiment, we use RGNN to identify patterns from the connectome that can effectively discriminate subjects by their gender or age characteristics. Following the observation in Ingalhalikar *et al.* (2014), we find that RGNN can effectively predict the meta-information based on the structural con-

Table 2: Gender/age prediction.

Methods	Gender	Age
DeepWalk	77.09% (± 2.46)	43.10% (± 3.85)
GCN	81.33% (± 2.36)	40.08% (±3.12)
FCN	89.80% (±2.94)	44.72% (±3.96)
RGNN	**92.75% (±1.70)**	**47.19% (±3.06)**

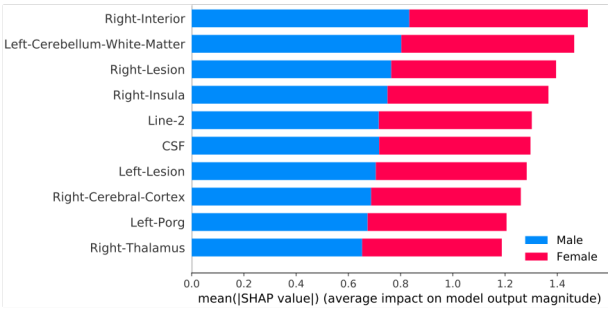


Figure 2: Node importances from SHAP analysis of age prediction model.

Figure 5.3: Left: Gender and Age Classification Results Obtained from Proposed Approach and the Baselines. Right: Shapley Values Highlighting the Importance of Different Regions in the Brain on Gender Classification.

nectome. Further, we perform sensitivity analysis (using SHAP), Figure 5.3 shows node-level importance, and so the regions corresponding to the *Right* part of the brain carry crucial information. Edge-level sensitivity analysis indicates that within hemispheric connectivities are of importance for gender prediction, and our results corroborate with the findings in Ingalhalikar *et al.* (2014). Similarly, with age prediction (Table 3), RGNN produces highly accurate predictions compared to baseline methods.

These two empirical studies clearly establish RGNN as a potential choice for processing structural and functional connectomes in order to perform more challenging tasks, such as predicting behavioral traits and cognitive states.

5.4 Summary

For biological graph data such as human brain connectome, many classical approaches have relied on hand-engineering statistical descriptors from structural or functional connectomes to build predictive models that are sub-optimal. We develop a structured network architecture termed (RGNN) that uses the connectome to con-

strain the message passing between two network layers representing edges and nodes, respectively (Chapter 5). Using connectomes from the Human Connectome Project (HCP), we show that the proposed approach can effectively predict meta-information such as age and gender and accurately recover the volumes of different brain regions, which are known to be encoded in the connectomes.

Chapter 6

CONCLUSIONS

Representation learning is critical to a wide range of applications that involve network analysis and inference. Machine learning models that can exploit the inherent structure in data have gained prominence in recent times. However, the main challenge in graph-based machine learning is that the structure of the graph/network is irregular when compared with images, or audio, or text. Networks are non-Euclidean and have complex topological structures. Simple operations like convolution, translation, and downsampling can not be explicitly defined on graphs. Despite the challenges, there is a surge in deep learning solutions for graph-structured data due to its widespread applicability in several fields. Graph attention networks (GAT), a recent addition to the broad class of feature learning models in graphs, utilize the attention mechanism to efficiently learn continuous vector representations for semi-supervised learning problems.

With the widespread adoption of attention models in language modeling and computer vision, it has become imperative to study and understand the functioning and robustness of attention mechanisms. Though GATs have successfully achieved state-of-the-art performance in semi-supervised node classification, a detailed analysis of the attention mechanism is not provided. In particular, the robustness of the attention mechanism in the presence of adversaries(for example, noisy nodes) needs to be studied. In this research area, we performed a detailed analysis of GAT models and presented exciting insights into their behavior. In particular, we showed that the models are vulnerable to adversaries (rogue nodes) and hence proposed novel regularization strategies to improve the robustness of GAT models. Using benchmark

datasets, we demonstrated performance improvements on semi-supervised learning using the proposed robust variant of GAT.

Graph Neural Networks (GNNs), a generalization of neural networks to graph-structured data, are often implemented using message passes between entities of a graph. While GNNs are effective for node classification, link prediction, and graph classification, they are vulnerable to adversarial attacks. GNNs inherits both advantages and disadvantages of DNNs. A small perturbation to the structure can lead to non-trivial performance degradation. Uncertainty Matching GNN (`UM-GNN`) architecture is proposed for improving the robustness of GNN models, particularly against poisoning attacks to the graph structure. `UM-GNN` leverages epistemic uncertainties from the message passing framework. More specifically, we employed a surrogate predictor that does not directly access the graph structure but systematically extracts reliable knowledge from a standard GNN through a novel uncertainty-matching strategy. Interestingly, this uncoupling makes `UM-GNN` immune to evasion attacks by design and achieves significantly improved robustness against poisoning attacks.

Even with the available GNN architectures, extending solutions from the single-graphs to multiplex graphs is not straightforward. During my research, attention models are developed for multi-layered graphs in semi-supervised learning problems. Two architectures GrAMME-SG and GrAMME-Fusion that exploit the inter-layer dependencies for building multi-layered graph embeddings were developed. Using empirical studies on several benchmark datasets, we evaluated the proposed approaches and demonstrated significant performance improvements in comparison to state-of-the-art network embedding strategies. The results also show that using simple random features is an effective choice, even when explicit node attributes are not available.

For biological data such as the human brain connectome, many classical approaches have relied on hand-engineering statistical descriptors from structural or

functional connectomes to build predictive models. However, there is growing interest in leveraging deep learning techniques. Though the human connectome is often viewed as a graph defined with each node indicating a brain region, and the edges representing neural connections, we argue that existing graph neural network solutions that are built on the assumption of information diffusion are not directly applicable. We developed a structured network architecture termed relational graph neural network (RGNN) that uses the connectome to constrain the message passing between two network layers representing edges and nodes. Using connectomes from the Human Connectome Project (HCP), we show that the proposed approach can effectively predict meta-information such as age and gender and accurately recover the volumes of different brain regions, which are known to be encoded in the connectomes.

6.1 Future Research Directions

Graph Neural Networks have become a de facto standard for representation learning on graphs. In the context of machine learning, GNNs are used in various supervised, semi-supervised, and even unsupervised tasks. GNNs can be readily applied to data that have explicit structural relationships. For non-structural data, where the relational structure is implicit or absent, an explicit graph structure can be constructed. In images and texts, dependency trees and scene graphs are usually constructed for advanced reasoning.

Many long-standing traditional problems can such as graph matching, can be addressed by employing GNNs. In physics, GNNs are used for particle state simulation (Kipf *et al.*, 2018). Any physical system or framework can be modeled as interconnected objects exhibiting pair-wise relationships between them. Classic recommendation problems can be framed as a link prediction problem between two sets of nodes in a bipartite graph. In the financial sector, GNNs are used for credit monitoring and

fraud detection.

In biology and medicinal chemistry, GNNs are exclusively used compared to other contemporary deep learning networks. Molecular fingerprints of a chemical molecule represented as 2d graphs can be obtained using GNNs. GNNs can uncover gene-disease association from gene expression data for a single cell. In protein-protein interactome (PPI) networks, many of the unknown drug-protein interactions, unknown drug-drug interactions can be predicted with the help of GNNs. In drug development and discovery, generative GNN models produce drug analogs for targeted proteins.

Ahmed, A., N. Shervashidze, S. Narayanamurthy, V. Josifovski and A. J. Smola, "Distributed large-scale natural graph factorization", in "Proceedings of the 22nd international conference on World Wide Web", pp. 37–48 (ACM, 2013).

Anirudh, R. and J. J. Thiagarajan, "Bootstrapping graph convolutional neural networks for autism spectrum disorder classification", in "ICASSP 2019", pp. 3197–3201 (2019).

Atwood, J. and D. Towsley, "Diffusion-convolutional neural networks", in "Advances in neural information processing systems", pp. 1993–2001 (2016).

Barone, A. V. M., J. Helcl, R. Sennrich, B. Haddow and A. Birch, "Deep architectures for neural machine translation", arXiv preprint arXiv:1707.07631 (2017).

Bazzi, M., M. A. Porter, S. Williams, M. McDonald, D. J. Fenn and S. D. Howison, "Community detection in temporal multilayer networks, with an application to correlation networks", Multiscale Modeling & Simulation **14**, 1, 1–41 (2016).

Belkin, M. and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation", Neural computation **15**, 6, 1373–1396 (2003).

Blundell, C., J. Cornebise, K. Kavukcuoglu and D. Wierstra, "Weight uncertainty in neural networks", arXiv preprint arXiv:1505.05424 (2015).

Boden, B., S. Günnemann, H. Hoffmann and T. Seidl, "Mining coherent subgraphs in multi-layer graphs with edge labels", in "Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 1258–1266 (ACM, 2012).

Bojchevski, A. and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning", in "International Conference on Machine Learning", pp. 695–704 (PMLR, 2019).

Bruna, J., W. Zaremba, A. Szlam and Y. LeCun, "Spectral networks and locally connected networks on graphs", arXiv preprint arXiv:1312.6203 (2013).

Cao, S., W. Lu and Q. Xu, "Grarep: Learning graph representations with global structural information", in "Proceedings of the 24th ACM international on conference on information and knowledge management", pp. 891–900 (2015).

Chakraborty, A., M. Alam, V. Dey, A. Chattopadhyay and D. Mukhopadhyay, "Adversarial attacks and defences: A survey", arXiv:1810.00069 (2018).

Chen, J., Y. Wu, X. Xu, Y. Chen, H. Zheng and Q. Xuan, "Fast gradient attack on network embedding", arXiv:1809.02797 (2018).

Chen, M., K. Kuzmin and B. K. Szymanski, "Community detection via maximization of modularity and its variants", IEEE Transactions on Computational Social Systems **1**, 1, 46–65 (2014).

Chen, P.-Y. and A. O. Hero, "Multilayer spectral graph clustering via convex layer aggregation: Theory and algorithms", IEEE Transactions on Signal and Information Processing over Networks **3**, 3, 553–567 (2017).

Dai, H., H. Li, T. Tian, X. Huang, L. Wang, J. Zhu and L. Song, "Adversarial attack on graph structured data", arXiv:1806.02371 (2018).

Defferrard, M., X. Bresson and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering", in "Advances in Neural Information Processing Systems", pp. 3844–3852 (2016a).

Defferrard, M., X. Bresson and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering", in "Advances in Neural Information Processing SystemsNIPS", pp. 3837–3845 (2016b).

Desikan, R. S., F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, M. S. Albert and R. J. Killiany, "An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest", Neuroimage **31**, 3, 968–980 (2006).

Dong, X., P. Frossard, P. Vandergheynst and N. Nefedov, "Clustering with multi-layer graphs: A spectral perspective", IEEE Transactions on Signal Processing **60**, 11, 5820–5831 (2012).

Dong, X., P. Frossard, P. Vandergheynst and N. Nefedov, "Clustering on multi-layer graphs via subspace analysis on grassmann manifolds", IEEE Transactions on signal processing **62**, 4, 905–918 (2014).

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

Eagle, N. and A. S. Pentland, "Reality mining: sensing complex social systems", Personal and ubiquitous computing **10**, 4, 255–268 (2006).

Feng, F., X. He, J. Tang and T.-S. Chua, "Graph adversarial training: Dynamically regularizing based on graph structure", IEEE Transactions on Knowledge and Data Engineering (2019).

Gal, Y. and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning", in "International Conference on Machine Learning", pp. 1050–1059 (2016).

Girvan, M. and M. E. Newman, "Community structure in social and biological networks", Proceedings of the national academy of sciences **99**, 12, 7821–7826 (2002).

Gligorijević, V., Y. Panagakis and S. Zafeiriou, "Fusion and community detection in multi-layer graphs", in "Pattern Recognition (ICPR), 2016 23rd International Conference on", pp. 1327–1332 (IEEE, 2016).

Goodfellow, I., Y. Bengio and A. Courville, *Deep learning* (MIT press, 2016).

Goodfellow, I. J., J. Shlens and C. Szegedy, "Explaining and harnessing adversarial examples", arXiv:1412.6572 (2014).

Grover, A. and J. Leskovec, "node2vec: Scalable feature learning for networks", in "Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining", pp. 855–864 (ACM, 2016).

Hackett, J. M., "Zachary's karate club", URL `https://studentwork.prattsi.org/infovis/labs/zacharys-karate-club/` (2019).

Hamilton, W., Z. Ying and J. Leskovec, "Inductive representation learning on large graphs", in "Advances in Neural Information Processing Systems", pp. 1024–1034 (2017).

Harris, Z. S., "Distributional structure", Word **10**, 2-3, 146–162 (1954).

Henaff, M., J. Bruna and Y. LeCun, "Deep convolutional networks on graph-structured data", arXiv preprint arXiv:1506.05163 (2015).

Henderson, K., B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos and L. Li, "Rolx: structural role extraction & mining in large graphs", in "Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 1231–1239 (ACM, 2012).

Ingalhalikar, M., A. Smith, D. Parker, T. D. Satterthwaite, M. A. Elliott, K. Ruparel, H. Hakonarson, R. E. Gur, R. C. Gur and R. Verma, "Sex differences in the structural connectome of the human brain", Proceedings of the National Academy of Sciences **111**, 2, 823–828 (2014).

Irwin, J. J., T. Sterling, M. M. Mysinger, E. S. Bolstad and R. G. Coleman, "Zinc: a free tool to discover chemistry for biology", Journal of chemical information and modeling **52**, 7, 1757–1768 (2012).

Jeurissen, B., J.-D. Tournier, T. Dhollander, A. Connelly and J. Sijbers, "Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data", Neuroimage **103**, 411–426 (2014).

Jin, M., H. Chang, W. Zhu and S. Sojoudi, "Power up! robust graph convolutional network against evasion attacks based on graph powering", arXiv:1905.10029 (2019).

Jin, W., Y. Li, H. Xu, Y. Wang and J. Tang, "Adversarial attacks and defenses on graphs: A review and empirical study", arXiv:2003.00653 (2020).

Keskar, K. A. A. N. S. and R. Socher, "Weighted transformer network for machine translation", CoRR **abs/1711.02132**, URL `http://arxiv.org/abs/1711.02132` (2017).

Kilbertus, N., M. R. Carulla, G. Parascandolo, M. Hardt, D. Janzing and B. Schölkopf, "Avoiding discrimination through causal reasoning", in "Advances in Neural Information Processing Systems", pp. 656–666 (2017).

Kim, J. and J.-G. Lee, "Community detection in multi-layer graphs: A survey", ACM SIGMOD Record **44**, 3, 37–48 (2015).

Kim, J., J.-G. Lee and S. Lim, "Differential flattening: A novel framework for community detection in multi-layer graphs", ACM Transactions on Intelligent Systems and Technology (TIST) **8**, 2, 1–23 (2016).

Kim, J., J.-G. Lee and S. Lim, "Differential flattening: A novel framework for community detection in multi-layer graphs", ACM Transactions on Intelligent Systems and Technology (TIST) **8**, 2, 27 (2017).

Kipf, T., E. Fetaya, K.-C. Wang, M. Welling and R. Zemel, "Neural relational inference for interacting systems", in "International Conference on Machine Learning", pp. 2688–2697 (PMLR, 2018).

Kipf, T. N. and M. Welling, "Semi-supervised classification with graph convolutional networks", arXiv preprint arXiv:1609.02907 (2016).

Kipf, T. N. and M. Welling, "Semi-supervised classification with graph convolutional networks", in "International Conference on Learning Representations (ICLR", (2017).

Li, J., C. Chen, H. Tong and H. Liu, "Multi-layered network embedding", in "Proceedings of the 2018 SIAM International Conference on Data Mining", pp. 684–692 (SIAM, 2018).

Liu, W., P.-Y. Chen, S. Yeung, T. Suzumura and L. Chen, "Principled multilayer network embedding", in "2017 IEEE International Conference on Data Mining Workshops (ICDMW)", pp. 134–141 (IEEE, 2017).

Liu, X., S. Si, X. Zhu, Y. Li and C. Hsieh, "A unified framework for data poisoning attack to graph-based semi-supervised learning", arXiv:1910.14147 (2019).

Lundberg, S. M. and S.-I. Lee, "A unified approach to interpreting model predictions", in "Advances in Neural Information Processing Systems", pp. 4765–4774 (2017).

Mucha, P. J., T. Richardson, K. Macon, M. A. Porter and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks", science **328**, 5980, 876–878 (2010).

Newman, M. E., "Finding community structure in networks using the eigenvectors of matrices", Physical review E **74**, 3, 036104 (2006).

Ng, A. Y., M. I. Jordan and Y. Weiss, "On spectral clustering: Analysis and an algorithm", in "Advances in neural information processing systems", pp. 849–856 (2002).

Niepert, M., M. Ahmed and K. Kutzkov, "Learning convolutional neural networks for graphs", in "International conference on machine learning", pp. 2014–2023 (2016).

Ou, M., P. Cui, J. Pei, Z. Zhang and W. Zhu, "Asymmetric transitivity preserving graph embedding", in "Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 1105–1114 (2016).

Papalexakis, E. E., L. Akoglu and D. Ience, "Do more views of a graph help? community detection and clustering in multi-graphs", in "Information fusion (FUSION), 2013 16th international conference on", pp. 899–905 (IEEE, 2013).

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

Ren, K., T. Zheng, Z. Qin and X. Liu, "Adversarial attacks and defenses in deep learning", Engineering (2020).

Santoro, A., D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia and T. Lillicrap, "A simple neural network module for relational reasoning", in "Advances in Neural Information Processing Systems 30", edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, pp. 4967–4976 (Curran Associates, Inc., 2017).

Schlimmer, J. C., "Concept acquisition through representational adjustment", (1987).

Sen, P., G. Namata, M. Bilgic, L. Getoor, B. Galligher and T. Eliassi-Rad, "Collective classification in network data", AI magazine **29**, 3, 93 (2008).

Shanthamallu, U., Q. Li, J. J. Thiagarajan, R. Anirudh and P. Bremer, "Modeling human brain connectomes using structured neural networks", Tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States) (2019a).

Shanthamallu, U. S., A. Spanias, C. Tepedelenlioglu and M. Stanley, "A brief survey of machine learning methods and their sensor and iot applications", in "2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)", pp. 1–8 (IEEE, 2017).

Shanthamallu, U. S., J. J. Thiagarajan, H. Song and A. Spanias, "Gramme: Semisupervised learning using multilayered graph attention models", IEEE transactions on neural networks and learning systems **31**, 10, 3977–3988 (2019b).

Shanthamallu, U. S., J. J. Thiagarajan and A. Spanias, "A regularized attention mechanism for graph attention networks", in "ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)", pp. 3372–3376 (2020).

Shanthamallu, U. S., J. J. Thiagarajan and A. Spanias, "A regularized attention mechanism for graph attention networks", in "ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)", pp. 3372–3376 (2020).

Shanthamallu, U. S., J. J. Thiagarajan and A. Spanias, "Uncertainty-matching graph neural networks to defend against poisoning attacks", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 35, pp. 9524–9532 (2021).

Shuman, D. I., S. K. Narang, P. Frossard, A. Ortega and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains", IEEE Signal Processing Magazine **30**, 3, 83–98 (2013).

Simon, P. L., M. Taylor and I. Z. Kiss, "Exact epidemic models on graphs using graph-automorphism driven lumping", Journal of mathematical biology **62**, 4, 479–508 (2011).

Smith, R. E., J.-D. Tournier, F. Calamante and A. Connelly, "Anatomically-constrained tractography: improved diffusion MRI streamlines tractography through effective use of anatomical information", Neuroimage **62**, 3, 1924–1938 (2012).

Smith, R. E., J.-D. Tournier, F. Calamante and A. Connelly, "SIFT2: Enabling dense quantitative assessment of brain white matter connectivity using streamlines tractography", Neuroimage **119**, 338–351 (2015).

Song, H., D. Rajan, J. J. Thiagarajan and A. Spanias, "Attend and diagnose: Clinical time series analysis using attention models", arXiv preprint arXiv:1711.03905 (2017).

Song, H. and J. J. Thiagarajan, "Improved community detection using deep embeddings from multi-layer graphs", arXiv preprint (2018).

Sporns, O., G. Tononi and R. Kötter, "The human connectome: a structural description of the human brain", PLoS Comput Biol **1**, 4, e42 (2005).

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", The journal of machine learning research **15**, 1, 1929–1958 (2014).

Sun, L., Y. Dou, C. Yang, J. Wang, P. S. Yu and B. Li, "Adversarial attack and defense on graph data: A survey", arXiv:1812.10528 (2018).

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, "Intriguing properties of neural networks", arXiv pre. arXiv:1312.6199 (2013).

Tagarelli, A., A. Amelio and F. Gullo, "Ensemble-based community detection in multilayer networks", Data Mining and Knowledge Discovery **31**, 5, 1506–1543 (2017).

Tang, J., M. Qu, M. Wang, M. Zhang, J. Yan and Q. Mei, "Line: Large-scale information network embedding", in "Proceedings of the 24th International Conference on World Wide Web", pp. 1067–1077 (International World Wide Web Conferences Steering Committee, 2015).

Tang, W., Z. Lu and I. S. Dhillon, "Clustering with multiple graphs", in "Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on", pp. 1016–1021 (IEEE, 2009).

Thiagarajan, J. J., P. Sattigeri, K. N. Ramamurthy and B. Kailkhura, "Robust local scaling using conditional quantiles of graph similarities", in "Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on", pp. 762–769 (IEEE, 2016).

Torng, W. and R. B. Altman, "Graph convolutional neural networks for predicting drug-target interactions", Journal of Chemical Information and Modeling **59**, 10, 4131–4149 (2019).

Tournier, J. D., F. Calamante and A. Connelly, "Improved probabilistic streamlines tractography by 2nd order integration over fibre orientation distributions", **18** (2010).

Tournier, J.-D., F. Calamante, D. G. Gadian and A. Connelly, "Direct estimation of the fiber orientation density function from diffusion-weighted MRI data using spherical deconvolution", Neuroimage **23**, 3, 1176–1185 (2004).

Van Essen, D. C., S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium *et al.*, "The wu-minn human connectome project: an overview", Neuroimage **80**, 62–79 (2013).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need", in "Advances in Neural Information Processing Systems", pp. 5998–6008 (2017).

Velickovic, P., G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, "Graph attention networks", arXiv preprint arXiv:1710.10903 (2017).

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, "Graph Attention Networks", International Conference on Learning Representations URL `https://openreview.net/forum?id=rJXMpikCZ` (2018).

Verma, V., M. Qu, A. Lamb, Y. Bengio, J. Kannala and J. Tang, "Graphmix: Regularized training of graph neural networks for semi-supervised learning", arXiv:1909.11715 (2019).

Vickers, M. and S. Chan, "Representing classroom social structure", Victoria Institute of Secondary Education, Melbourne (1981).

Wang, M., L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A. J. Smola and Z. Zhang, "Deep graph library: Towards efficient and scalable deep learning on graphs", ICLR Workshop on Representation Learning on Graphs and Manifolds URL `https://arxiv.org/abs/1909.01315` (2019).

Waniek, M., T. P. Michalak, M. J. Wooldridge and T. Rahwan, "Hiding individuals and communities in a social network", Nature Human Behaviour **2**, 2, 139–147 (2018).

Wu, H., C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu and L. Zhu, "Adversarial examples on graph data: Deep insights into attack and defense", arXiv:1903.01610 (2019a).

Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang and S. Y. Philip, "A comprehensive survey on graph neural networks", IEEE Transactions on Neural Networks and Learning Systems (2020).

Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, "A comprehensive survey on graph neural networks", arXiv preprint arXiv:1901.00596 (2019b).

Xu, K., H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective", arXiv:1906.04214 (2019).

Yang, L., X. Cao, D. He, C. Wang, X. Wang and W. Zhang, "Modularity based community detection with deep learning.", in "IJCAI", pp. 2252–2258 (2016).

Yang, Z., W. Chen, F. Wang and B. Xu, "Improving neural machine translation with conditional sequence generative adversarial nets", arXiv preprint arXiv:1703.04887 (2017).

Zeng, Z., J. Wang, L. Zhou and G. Karypis, "Coherent closed quasi-clique discovery from large dense graph databases", in "Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining", pp. 797–802 (ACM, 2006).

Zhang, H., L. Qiu, L. Yi and Y. Song, "Scalable multiplex network embedding.", in "IJCAI", pp. 3082–3088 (2018a).

Zhang, Z., G. I. Allen, H. Zhu and D. Dunson, "Relationships between human brain structural connectomes and traits", bioRxiv URL `https://www.biorxiv.org/content/early/2018/01/31/256933` (2018b).

Zhu, D., Z. Zhang, P. Cui and W. Zhu, "Robust graph convolutional networks against adversarial attacks", in "Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining", pp. 1399–1407 (2019).

Zügner, D., A. Akbarnejad and S. Günnemann, "Adversarial attacks on neural networks for graph data", in "Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining", pp. 2847–2856 (2018).

Zügner, D. and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning", arXiv:1902.08412 (2019).

BIOGRAPHICAL SKETCH

Uday Shankar Shanthamallu is currently pursuing his Ph.D. degree with the school of Electrical, Computer and Energy Engineering at Arizona State University. He received his Master's degree in electrical engineering from Arizona State University (ASU) in 2018 and a Bachelor's degree in electronics and communication engineering from the National Institute of Engineering, India, in 2011. His research interests include representation learning for graphs using machine learning and deep learning techniques. He also has experience on sensor data analytics for anomaly detection. His internship with NXP Semiconductors (2016) focused on algorithm development for sensor data analytics. He also interned with Lawrence Livermore National Laboratory during the summer of 2019 and 2020 where he built predictive models for human brain connectomes.